Light Morphology and Arabic Information Retrieval

Mohammed Algarni

Department of Computer Science and Software Engineering

University of Canterbury

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

September 2016

Supervisory Committee:

Professor Tim Bell , Senior Supervisor

Dr. Kourosh Neshatian , Associate Supervisor

بِسمِ اللَّهِ الرَّحمنِ الرَّحيم

"الحَمدُ لِلَّهِ الذي أَنزَلَ عَلَى عَبدِهِ الكِتَابَ وَلَم يَجعَل لَهُ عِوَجَا"

صَدَقَ اللَّهُ العَظِيم

وَالصَّلاةُ وَالسَّلامُ على سَيِّدِنَا مُحَمَّدٍ وعَلَى آلِهِ وصَحبِهِ أَجمَعِين.

To my father and my mother...

رَبِّ ارحَمهُمَا كَمَا رَبَّيَانِي صَغِيرَا.

Acknowledgements

Thank God this thesis has been completed, only by His Grace. All words of gratitude would not be enough to express my appreciation to my senior supervisor Professor Tim Bell. He has given me the time and thoughts any PhD student can wish for. I have benefited great deal from the discussions with Professor Bell. It was because of his insights and ideas that I managed to publish a paper in the ACM CIKM'14, and complete this thesis. I also would like to speak highly of, and I'm sure many people share this feeling with me, his down to earth attitude, which made me feel important and worthy of expressing my ideas because they could mean something to someone. I was always under the impression that Professor Bell really wanted me to think more and maybe find some ideas here or there, this feeling was the force beyond my striving to finish my PhD course. I will always be in debt to Professor Bell for that.

I owe Dr. Brent Martin a lot of credit for giving me the guidance I needed when I first joined Canterbury. He showed me the way to conduct a solid academic research, it was because of Dr. Martin's teaching that I have been able to finish this research. Furthermore, I would like to thank him for making me feel at home when I first arrived at Christchurch.

I would like to express my deep appreciation to Dr. Kourosh Neshatian, my associate supervisor, for his time and ideas. He joined the supervisory team at a crucial time of my research, I thank him for his smart comments and feedback. He always had the time and offered his best in our meetings. Dr. Neshatian gave me such great ideas on how to enhance the paper, it was accepted with no changes necessary for the final version.

Credit is due to two of my best friends: Eisa Abdullah Aleisa and Ali Salem Dughsan Alghamdi. I thank Eisa for his support, and also for encouraging me to pursuit my study in the field of Arabic Information Retrieval (AIR). Eisa also suggested Christchurch, New Zealand as the place of study, and I am really grateful for that in particular because I have spent some of my best years in this beautiful town amongst its lovely people.

Ali Salem deserves many thanks for his relentless support all the way through till the end, he was generous enough to buy a brand new Macbook Pro and gave it to me as a gift when I decided to join Canterbury. I conducted the research using this nice machine, and I still enjoy using it up to now. Ali Salem believed in me and I am grateful for that.

Last but not least, my wonderful wife Umm Abdallah, who saw the whole journey through, is entitled for my love and gratitude so long as I live. She struggled as hard as I did, but the difference was that she never complained. I have always come back home from the university and found a very nice and warm welcome, she really was so sincere that I felt I was doing the PhD for both of us.

Abstract

The chief purpose of this study is to investigate the impact of morphology on Arabic Information Retrieval (AIR). In doing so, different forms of the surface word have to be examined as indexing terms in order to learn which is the most effective in performance. Experiments are needed starting with the root all the way to the surface form so that we can evaluate the difference each selection makes. This has resulted in the development of two experimental stemmers for the Modern Standard Arabic (MSA), one light and the other root-based, which will be referred to hereafter as the Simple Arabic Stemmer (SAS). The stemmers were based on the Quran morphology and constructed according to its rules. They conform to the Quran guidelines in terms of segmenting a word into its correct morphological combination (prefix-pattern-suffix). The reason for leveraging the Quran as a morphological knowledge base was that the Arabic morphological rules were documented according to the Quran relatively soon after it became known. Using the Text REtrieval Conference (TREC) 2002 Arabic corpus, which contains 383,872 documents, 75 topics, and 10,031 manually-judged documents, we test our approach against two widely-used root stemmers, Khoja and Sebawai. In the experiments, our root algorithm has generated better Mean Average Precision (MAP), giving a 13% relative gain over the other stemmers. The Simple Arabic Stemmer outperformed both stemmers in producing more accurate roots for the TREC corpus. We demonstrated that, by placing a restriction on what prefix-pattern-suffix combinations are permissible on the surface, the stemming process would be enhanced, and fewer stemming errors are produced. Another experiment was conducted to measure the difference between the stem and the root

as indexing terms. Due to the fact that a root conflates so many stems under one form, its precision degraded when used as an indexing term. The results obtained favoured choosing the stem as an indexing term.

Contents

Contents vi

List of Figures x

List of Tables xii

Nomenclature xvi

1 Introduction 1

1.1 The Arabic Language . . . 4
1.2 The Arabic Language Distinct Properties . . . 6
1.2.1 Root-Pattern Morphology . . . 7
1.2.2 Wealth of Inflectional Operations . . . 10
1.2.3 Non-concatenative Morphology . . . 11
1.3 Arabic Morphology and IR . . . 12
1.4 Research Questions and Goals . . . 14
1.5 Organisation . . . 15

2 Preliminaries 16

2.1 Orthography . . . 18
2.2 Phonology . . . 22
2.3 Morphology . . . 25
2.3.1 Derivation . . . 26
2.3.1.1 Roots . . . 26
2.3.1.2 Patterns . . . 28
2.3.2 Inflection . . . 33

2.3.2.1 Prefixes 33
2.3.2.2 Suffixes 35
2.4 Conclusion 36

3 Current Stemmers 38
3.1 Overview 40
3.2 Root Stemmers 42
3.2.1 Xerox Arabic Morphological Analyzer 42
3.2.2 Khoja 46
3.2.3 Sebawai 50
3.3 Light Stemmers 52
3.3.1 BAMA 52
3.3.1.1 Segment Word 54
3.3.1.2 Analyze 56
3.3.2 Light10 58
3.3.2.1 stemPrefix 59
3.3.2.2 stemSuffix 60
3.3.3 MADAMIRA 62
3.4 Conclusion 63

4 Enhancing Arabic Stemming via Morphological Analysis 64
4.1 Haifa 65
4.2 Leeds 68
4.3 Simple Stemmer Customised Methodology 71
4.4 Simple Arabic Morphological Lexicon 78
4.4.1 Roots 79
4.4.2 Nominal System 80
4.4.2.1 Prefixes 81
4.4.2.2 Suffixes 83
4.4.2.3 Patterns 84
4.4.2.4 Legal Combinations 94
4.4.3 Verbal System 96
4.4.3.1 Prefixes 97

4.4.3.2 Suffixes 98
4.4.3.3 Patterns 98
4.4.3.4 Legal Combinations 99
4.5 SAS Implementation Overview 100
4.5.1 SAS Major Classes and Functions 100
4.5.2 Integrating SAS into Lucene 106
4.6 Conclusion 110

5 Evaluation **111**
5.1 Standard IR Evaluation Methods 112
5.2 TREC Collection 112
5.2.1 TREC Documents 113
5.2.2 TREC Topics or Queries 114
5.2.3 TREC Relevance Judgments 115
5.3 The ZAD Collection 115
5.3.1 ZAD Documents, Queries, Relevance Judgments 116
5.4 Common Evaluation Measures 117
5.5 Work Evaluating AIR Prior to TREC 119
5.6 TREC Experiments 120
5.7 The Evaluation of the Root as an Indexing Term 122
5.7.1 Root vs. Stem 123
5.7.2 Khoja vs. Sebawai vs. SAS 126
5.7.3 Issues in Arabic Root Stemming 131
5.7.4 Suggested Solutions 136
5.8 The Evaluation of the Stem as an Indexing Term 138
5.8.1 Issues in Arabic Light Stemming 143
5.9 Conclusion 149

6 Conclusion and Future Directions **150**
6.1 Contributions 151
6.2 Limitations 153
6.3 Future Directions 154
6.4 Concluding Remarks 155

References 157

List of Figures

1.1 Searching with Inflected Arabic Words. 3
1.2 Searching with Stemmed Arabic Words. 3
1.3 Arabic Word Generation Process. 8

2.1 Long Vowels Phonological Nature Example. 20

3.1 Xerox Analyser Step-by-Step Example. 44
3.2 Buckwalter Arabic Morphological Analyser Main Steps. 53

4.1 Haifa Lexicon Construction Methodology. 66
4.2 Haifa Word's Morphological Record. 68
4.3 Leeds Quran Corpus Construction Methodology. 69
4.4 Leeds Word's Morphological Record. 70
4.5 Customised Lexicon Construction Methodology. 72
4.6 SAS Word's Morphological Record. 77
4.7 Proposed Solution Components. 78
4.8 Prefix-Pattern-Suffix Table Example. 95

5.1 TREC Arabic Document Example. 113
5.2 TREC English/Arabic Topic Example. 114
5.3 ZAD Document and Query Sample. 116
5.4 TREC Root vs. Stem Average Precision/Recall. 125
5.5 ZAD Collection Root vs. Stem Precision/Recall. 126
5.6 Khoja vs. Sebawai vs SAS Average Precision/Recall Graph. . . . 129
5.7 Khoja vs. Sebawai vs SAS Average Precision/Recall for ZAD. . . 130
5.8 Average Precision and Recall for the Stem Experiment. 142

5.9 Average Precision/Recall for the Stem Experiments on ZAD. . . . 143

List of Tables

1.1 Stemming Example 1
1.2 Examples of the Root-Pattern Morphology 8
1.2 Examples of the Root-Pattern Morphology 9
1.3 Dual and Plural Case-Marking Suffixes 10
1.4 Verbal Conjugation Example 11
1.5 Sound or Regular Plurals vs. Broken Plurals 12

2.1 The Arabic Alphabet Letters 18
2.1 The Arabic Alphabet Letters 19
2.2 Nominal Suffixes 21
2.3 Diacritics on the Consonant “b” 22
2.4 Vowels Surface Shapes 23
2.5 The Lengthening Rule 23
2.6 Vowel-toYeh-With-Hamza-Above Transformation 24
2.7 Vowel-to-Hamza Transformation 24
2.8 The Root “qwl” (قول) and its Combinations 27
2.9 Triliteral Verbal Pattern (Form I) 28
2.10 Quadriliteral Verbal Pattern Example 29
2.11 Triliteral Active Participle (Form I) 29

2.12 Active Participles for the Quadriliteral Verbal Pattern 29
2.13 Form I Passive Participle 30
2.14 Form II Passive Participle 30
2.15 Infinitive for the Triliteral Verbal Form I 30
2.16 Infinitives for the Quadrilateral Verbal Form 31
2.17 Verbal Adjectives 31
2.18 Intensive Adjectives 31
2.19 Triliteral Noun of Instrument 32
2.20 A Popular Triliteral Broken Plural Pattern 32
2.21 A Popular Quadriliteral Broken Plural Pattern 33
2.22 Nominal Prefixes Categories 34
2.23 Imperfective Conjugation Prefixes 35
2.24 Possessive Pronouns 36
2.25 Perfective Conjugation 36

3.1 Ambiguity with Full Vocalization 39
3.2 Current Arabic Stemmers Features and Limitations 41
3.3 Current Arabic Stemmers Lexical Comparison 42
3.4 BAMA Segments for the Word "wAldyn" 54

4.1 Operation of the Two Grammar Constraints, Digram Uniqueness and Rule Utility 73
4.2 A Sample Output after Running SEQUITUR on the Quran text 74
4.3 Segmenting Template Example 76
4.4 Quran Roots and their Category Distribution 79
4.5 The Prefix "l" and its Usage in the Quran 81
4.6 Four Prefixes Grouped in One Word 82
4.7 Rare Prefixes Grouping 82
4.8 Nominal Popular Prefixes 83
4.9 Nominal Popular Suffixes 84
4.10 Active Participles for the 10 Triliteral Verbal Patterns 85
4.11 Form I Active Participle Roots Distribution 85
4.12 Active Participles for the 2 Quadriliteral Verbal Patterns 86
4.13 Form I Passive Participle 86

4.14 Form II Passive Participle 86
4.15 Infinitives for the 10 Triliteral Verbal Forms 87
4.16 Infinitives for the Triliteral Verbal Forms 87
4.17 Infinitives for the 2 Quadrilateral Verbal Forms 88
4.18 Verbal Adjectives 88
4.19 Intensive Adjectives 89
4.20 Triliteral Nouns of Instrument 89
4.21 Triliteral Individuality Nouns 89
4.22 Quadriliteral Individuality Nouns 90
4.23 Triliteral Noun of Preeminence 90
4.24 A Singular Pattern with Different Broken Plural Patterns 91
4.25 Popular Triliteral Broken Plural Patterns 92
4.26 Popular Quadriliteral Broken Plurals Patterns 92
4.27 Singular Stems for the Broken Plural Pattern >CCAC 93
4.28 The Roots Distribution over the >CCAC Pattern 93
4.29 Nominal Prefix-Suffix Combinations 96
4.30 Verbal Popular Prefixes 97
4.31 Verbal Popular Suffixes 98
4.32 Triliteral Verbal Patterns 99
4.33 Common Quadriliteral Verbal Patterns 99
4.34 Verbal Prefix-Suffix Combinations 100

5.1 Micro-AIRS Experimental Results 120
5.2 The most Frequent Corpus Prefixes Found By UC 121
5.3 Al-Stem vs. the University of California Light Stemmer 121
5.4 Mean Average Precision 122
5.5 Mean Average Precision 123
5.6 Mean Average Precisions for ZAD Collection 123
5.7 Wilcoxon Signed-Rank Test 124
5.8 Precisions for Top Ranked Retrieved Documents 124
5.9 Precisions for Top Ranked Retrieved Documents for ZAD Experiments 125
5.10 Mean Average Precision 127

5.11 ZAD Collection Mean Average Precisions 127
5.12 Precisions at Highest Ranked Documents 128
5.13 Precisions at Highest Ranked Documents for ZAD 128
5.14 Correct Solution and Error Types 132
5.15 Light Stem Mean Average Precisions 139
5.16 Light Stem Mean Average Precision for ZAD Collection 140
5.17 Precisions for Top 20 Retrieved Documents for the TREC Collection 140
5.18 Precisions for Top 20 Retrieved Documents for the ZAD Collection 141
5.19 Wilcoxon Signed-Rank Test 141
5.20 The Five Stemmers' Performance Using 300 Random ZAD Words 143
5.21 Types of Errors for MADA Stemming 144
5.22 Types of Errors for AMIRA Stemming 145
5.23 Types of Errors for SAS Stemming 146
5.24 Types of Errors for AlStem Stemming 147
5.25 Types of Errors for Light10 Stemming 148

6.1 Mean Average Precision 152
6.2 Lexicon Numerical Comparison 152

Nomenclature

Accusative: When a noun is an object of a sentence or an adverb are two grammatical positions where it could carry an accusative case

Ar. Pat.: Arabic pattern "Alwazn" of the word itself, so the Ar. Pat. for the word "mdAris" "*schools*" is (مفَاعِل)

Ar. Word: The word, which is written in Buck., is also written in Arabic script

Buckwalter Transliteration: (Buck. for short) is a transliteration scheme that was created by Tim Buckwalter as part of his Arabic Morphological Analyser, where each Arabic letter is mapped to an ASCII character

Buckwalter Arabic Morphological Analyzer: (BAMA for short) is the first Arabic light stemmer used widely by many researchers before the TREC corpus existed, and was developed by Timothy Buckwalter after he left Xerox team, which built the Xerox Arabic Morphological Analyzer [17]

Case: A morphological feature needed for resolving grammatical analysis for the word, it is signified by short vowels or letters appended as the last suffix on the surface word. A case could be nominative, accusative or genitive

Dual: A morphological feature designating two subjects

Feminine: (F. for short) is a morphological feature designating a female subject

Genitive: When a noun is preceded by a prepositional particle for instance

Gloss: A column in each table giving more glossary information about the concerned item

Masculine: (M. for short) is a morphological feature designating a male subject

Nominative: When a noun is positioned at the beginning of a sentence, or comes as a subject of verb, it ought to be in a nominative case

Part Of Speech: (POS for short) is a feature of the word indicating wether it is a noun, a verb, or a particle

Pattern: A CV-template dictating how the radicals, the short vowels, and the derivational affixes are ordered to generate the stem

Plural: A morphological feature designating more than two subjects

Quran Index: (QI for short) consists of three integers separated by dots, which represent the **chapter**, the **verse** in which the word is mentioned, and the **word**'s number respectivelly

Root: Smallest lexical unit that can bear a meaning, and from which other words are derived using different patterns

Singular: (Sing. for short) is a morphological feature designating a single subject, whether be it a person or otherwise

Stem: Interdigitsing of the root and the pattern to generate the surface word before the addition of any affixes

TREC: **T**ext **RE**trieval **C**onference supports research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies

Unconn.: Unconnected Arabic letter written by itself with no preceding or following letter

Chapter 1

Introduction

In today's digital world, many users on the Internet need to retrieve information from the huge volume of data available. As the Internet offers additional textual resources everyday, it has become the medium for information consumers. Using only a few words, or a query as it will be called hereafter, the users aim to fetch all documents relevant to their search terms. The accuracy of the search engine depends on the query's words, therefore if the words were not precise enough, they might influence the search engine's ability to retrieve the target documents. That is why the words in the query ought to be decomposed into meaningful components before they are submitted to the retrieval system in order to ensure a satisfying level of precision and recall. This process is called stemming and it is employed by all search engines for the majority of the languages. Stemming is advantageous in information retrieval because it conflates many related terms into one indexing term, so it saves storage and look-up time.

Table 1.1: Stemming Example

Document Words	Index
Connected Connecting Connection Connections Connectivity	Connect

For example, according to Porter's stemmer, all the terms in the left column of Table 1.1 are indexed under "connect" [62].

In this thesis we will look at Arabic queries, we will use Buckwalter's transliteration characters to write Arabic words illustrating different morphological aspects of the Arabic language . This transliteration mapping was created by Tim Buckwalter as part of his Arabic Morphological Analyser, where each Arabic letter is represented by an ASCII character (e.g., the first letter is represented by ">") [17]. Buckwalter's transliteration was utilised by many institutes including the Linguistic Data Consortium (LDC), which owns and distributes the TREC Arabic corpus.[1]

The Arabic scripts will be displayed using a LaTeXpackage called ArabTeX, which displays Arabic letters on the surface in an elegant style [42]. Furthermore, illustrative words used in the examples hereafter would have their root's radicals distinguished by this colour, while the affixes will be given this colour. For instance, if we to rewrite the word "wAlxlylAn" in Figure 1.1, it would be of the form "wAlxlylAn" to show that the root is "xll", and "wAl" and "An" represent the prefix and the suffix respectively.

Arabic Information Retrieval (AIR) on the internet is way below the acceptable level in terms of precision. For instance, Google does not have the means to stem Arabic words, so it treats them as strings of unrelated sequences of characters.

Consider the words in Figure 1.1 where we tried to search for the two terms " والخليلان المتجافيان" (translated to the English alphabet as "wAlxlylAn AlmtjAfyAn") which means "*and the two restrained friends*", but the search produced no result.

[1] For the complete Buckwalter Arabic Lexicography, see http://www.qamus.org.

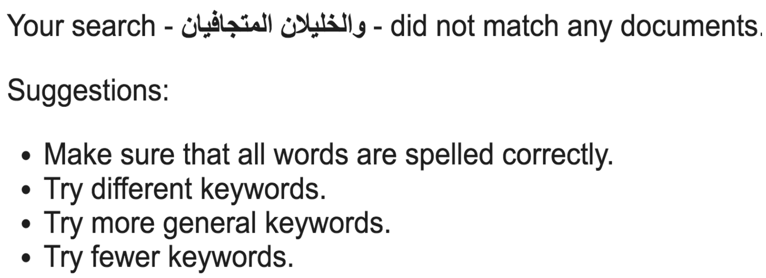


Figure 1.1: Searching with Inflected Arabic Words.

Figure 1.2 shows the result of the search for the same terms but after stemming the two words. The stemming process merely stripped off the prefixes "وال" ("wAl") from the first and "ال" ("Al") from the second, and the suffix "ان" ("An") from both words. The two words become "خليل متجافي" ("xlyl mtjAfy") meaning "*a restrained friend*", which produced 1,380 hits.

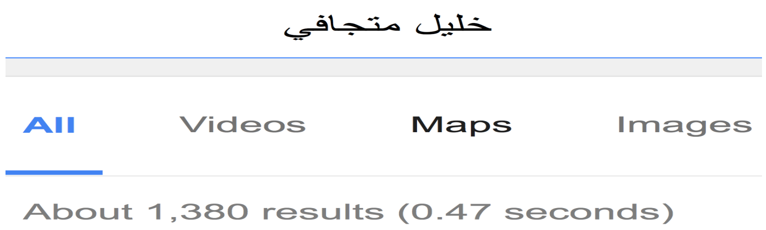


Figure 1.2: Searching with Stemmed Arabic Words.

It is fair to assume that the majority of the search engines on the internet do not utilise the Arabic morphology as effectively as done with other languages because:

- The complexity of the Arabic morphology means there are many derivational and inflectional operations, and these are more complex than English rules,

- The Arabic language has not been given much attention until recently [23; 28; 61; 74].

A simple form of the AIR problem is: given a user's query written in Modern Standard Arabic (MSA), retrieve all documents matching the query's terms. This is the basic boolean model of Informational Retrieval (IR); a deeper task is to locate all documents that are related to the query's terms, or belong to the same word class (sharing the same stem or root), thus opening the door for a new look at how the query's terms are internally structured.

In this study, we attempt to develop an Arabic root-finding and light-stemming algorithms with a concise lexicon, referred to hereafter as SAS root and SAS stem respectively. These two algorithms have been developed by decomposing the Quran's words and learning their internal morphological units, could be used in Information Retrieval (IR) to enhance its performance. In addition, we try to answer some key questions pertaining to AIR performance. For instance, when indexing Arabic words, which form (surface, stem, or root) is the most efficient?

We conduct experiments to measure the impact the selection of a particular indexing term would have on the precision of AIR. We implemented a Quran-based model that dictates certain restrictions on the stemming process by limiting the prefix-suffix combinations for each pattern to those allowed in the Quran. Then the lexicon was extended to include more MSA legal combinations by stemming 6,000 extra words collected from the Saudi newspaper Alriyadh[2] and the Lebanese newspaper Alnahar web archives.[3]

1.1 The Arabic Language

Good progress has been made with regard to other languages' text processing, for example in the English language, the level of automation has reached almost an

[2] http://www.alriyadh.com.
[3] http://www.annahar.com.

error-free status, the situation is different for Arabic. The Arabic text processing field is still in need of more research before it could offer any contribution to a larger framework such as IR, and the Arabic language lacks adequate resources and solutions tailored towards the language specifics.

There have been recently great advancements in modelling the Arabic morphology formally that have assisted computational linguists to implement it on machines, despite its complexity. McCarthy's analysis of Arabic morphological lexicon, where he segmented the lexicon into three layers; the root, the short vowels, and the pattern [50; 52]. His work was embraced widely by many morphologists because of its simplicity [39]. McCarthy's prosodic auto-segmental theory of non-concatenative morphology states that a word in the Arabic language is composed of three components:

- **Root:** the only morphological unit that can bear a meaning, and from which words can be derived using pattern templates. The root's letters are referred to as radicals and represented in the pattern CV-template as Cs.

- **Vocalism:** three short vowels; "a", "u" and "i", which are employed to give the root's radicals their sounds and represented in the pattern CV-template as Vs.

- **Pattern:** a CV-template dictating how root's radicals, short vowels, and derivational affixes are ordered to generate the stem.

The vowels include short and long vowels; short vowels are small marks (diacritics), usually written either above or underneath the letter. Moreover, short vowels are used to indicate grammatical cases for, verbs with singular subjects, singular nominal stems, and broken plurals.

The long vowels on the other hand, are three alphabet letters ("A", "w", and "y"), and are called the "weak letters" (حروف العلّة) by Arabic grammarians.

These three long vowels are the keys when deriving new words as they represent the exceptions of the derivational rules, and as a result pose an extra challenge for root stemming because of their phonological nature. They take different shapes on the surface depending on the presence of another long vowel in the pattern template, their position within the pattern template slots, and the short vowels before and after them, this phenomenon is clearly demonstrated in Figure 2.1.

In actuality, if it were not for the weak letters in the Arabic phonology, the Arabic stemming would be a simple rule-based process [14]. That is, the sound letters (the other 25 letters) appear on the surface intact no matter where they are placed in the pattern template, which makes extracting the root and other morphological features a task of pattern matching.

1.2 The Arabic Language Distinct Properties

The Arabic language presents one of the most challenging morphologies in Natural Language Processing (NLP) [4; 32]. It descends from the Semitic family with many inflectional and derivational morphological operations. What distinguishes Semitic languages from others is the mechanism by which words are generated. They follow a systematic derivational process that consists of root's radicals, short vowels, and derivational affixes arranged in a precise sequence known as the "*pattern*" [50].

An Arabic word must bear a root and a pattern within its morphological structure, or else it is a borrowed word. Most of the Arabic roots consist of three radicals, and "all students of Semitic are familiar with so-called principle of triplicity of consonants. This is to say that the vast majority of all Semitic stems are composed of three root consonants, or radicals. By consonant, I mean to include the obstruents, liquids, and glides" [14].

There are fundamental differences between the Arabic and the English language with regard to how words are composed. While the English language inflectional operations are confined to person and number (e.g., I write → he writes, school → schools), the Arabic language offers more operations including, besides person and number, gender, case-marking to designate the grammatical position of the word within the sentence, and imperfective verb moods. Those morphological features are usually marked by; suffixes for nouns, and both prefixes and suffixes for verbs.

Furthermore, the English language number category contains only two elements; namely the singular and the plural, whereas the Arabic language has an extra element used for marking the dual number feature. The dual number feature indicates that the subjects are two in number, and it makes no difference wether these two subjects are human or otherwise. Please see the Nomenclature on page xi for more information.

In the next three sections, we go through the distinguishing features of the Arabic Morphology in an attempt to show the reader why the Arabic language is much more complicated than the English language. Firstly, we explain the derivational process in more detail; or how morphological units are grouped to produce new words. We shed light on what is called, root-pattern morphology, using illustrative examples borrowed from the nominal system.

Secondly, we leverage the verbal system to delineate what is meant by a highly-inflective when referring to the Arabic language. Lastly, the Arabic language has a non-concatenative morphology, which is manifested very clearly in the Broken Plurals (BPs) category. We present different examples to show how the plural form is generated through a transformation process that is unique to the Arabic language, and as of yet poses a challenge for Arabic computational linguists.

1.2.1 Root-Pattern Morphology

To illustrate the process of generating words in the Arabic language, the following terms are defined as:

- **Stem:** a word generated by inserting the root's radicals and the short vowels into their respective slots on the pattern template.

- **Surface Word:** a stem with the addition of prefixes and suffixes if needed, or in short an inflected stem.

Consider Figure 1.3, where the root is "slm" ("having to do with submissiveness and safety") [14]. The Cs here represent the alphabet letters including the three long vowels (>, w, y), whereas the Vs represent the short vowels (a, u, i).

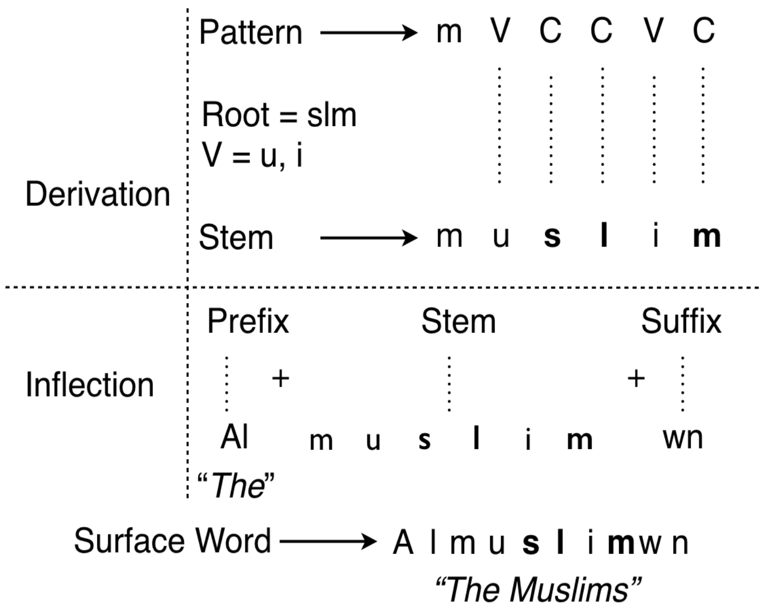


Figure 1.3: Arabic Word Generation Process.

There are two operations performed in this example: the first is the derivational which results in a stem with a new meaning produced by inserting the root's radicals and the short vowels into their respective slots in the pattern template.[9] The second is inflectional which is used merely to mark the number, gender and case endings for that particular stem, or give it its surface form without altering its meaning. There are many more stems that can be generated using the same root with different patterns.

Table 1.2: Examples of the Root-Pattern Morphology

Pattern	Word	Arabic	QI	Gloss
CaCiCa	salima	سَلِمَ		he was safe
CaC~aCa	sal~ama	سَلَّمَ	8.43.16	he protected
CACaCa	sAlama	سالم		he made up
>aCCaCa	>aslama	أسلم	3.83.6	became a Muslim

[9]The pattern in Figure 1.3 is the Active Participle of the verbal pattern IV, which is ">slama" (أسلم) "*became a Muslim*".

Table 1.2: Examples of the Root-Pattern Morphology

Pattern	Word	Arabic	QI	Gloss
taCaC~aCa	tasal~am	تَسَلَّمَ		he received
AiCtaCaCa	Aistalama	اِستَلَمَ		he was handed
AistaCCaCa	Aistaslama	اِستَسلَمَ		he surrendered
CACiC	sAlimwn	سَالِمون	68.43.11	safe
CiCC	Alsilmi	السِلمِ	2.208.6	peace
<iCCAC	Al<islAm	الإسلَام	61.7.11	submission (Islam)
muCCiC	muslimA	مُسلِمَا	3.67.10	muslim
CaCAC	salAm	سَلَام	6.54.7	greeting, soundness
CaCyC	salym	سَلِيم	26.89.6	sound or flawless
mustaCCiC	mustaslimwn	مُستَسلِمون	37.26.4	who gives up

Table 1.2 shows seven verbal and seven nominal stems, which have been all derived from the same root "slm". So the concluding remark is that "Due to Arabic's morphological complexity, Arabic retrieval benefits greatly from morphological analysis — particularly stemming" [21].

Depending on the position of the word within the sentence, the Arabic word could have one of three cases. A noun at the beginning of a sentence or a subject of verb ought to be in a nominative case with the suffix "u" (ـُ) for singulars, "Ani, tAni" (انِ، تَانِ) for masculine and feminine dual respectively, and "Atu" (اتُ) for the feminine or "wna" (ونَ) for the masculine plurals.

The accusative case for singulars is designated by the short vowel "a" (ـَ), whereas the genitive case is marked by the short vowel "i" (ـِ); the only short vowel written underneath the last letter of the word. On the other hand, the accusative and genitive suffixes for the dual masculine nominal stems are the same, namely "yni" (يْنِ), and "tayni" (تَيْنِ) for the feminine gender. Likewise, for the plural masculine nominal stems, the suffix "yina" (ينَ) signifies the accusative and the genitive cases, and the suffix "Ati" (اتِ) does the same for the feminine plurals.

Table 1.3: Dual and Plural Case-Marking Suffixes

Case	Dual	Plural
Nominative	"Ani, tAni" (انِ، تَانِ)	"Atu, wna" (اتُ، وُن)
Accusative	"yni, tyni" (ينِ، تَينِ)	"Ati, yina" (اتِ، ينَ)
Genitive	"yni, tyni" (ينِ، تَينِ)	"Ati, yina" (اتِ، ينَ)

Table 1.3 shows those cases, which are imperative in the grammatical analysis step but bear no value for Information Retrieval because key words are processed separate from their context. For example, "Almuslimwn" is the surface word, which contains besides the stem "muslim", the definiteness marker "Al" as a prefix, and the suffix "wn", which denotes the masculine nominative case.

1.2.2 Wealth of Inflectional Operations

In addition to the many derivational operations, the Arabic language is also considered a highly inflective language. This is to say that there are so many inflectional operations that ought to be performed to mark person, number, gender, and one extra feature pertaining to verbs; namely the mood feature.

To illustrate the richness of the Arabic verbal inflectional system, consider the Form I perfective verb, with the stem "katab" (كَتَب) [22]. The root is "ktb" (كتب) "*having to do with writing*", and the pattern is "CaCaC". The perfective form is the same as the English past tense verb, so the perfective Form I for the root "ktb" and a 3rd masculine singular subject is "kataba" (كَتَبَ) "*he wrote*".

It is to be noted that "the perfective conjugation requires the set of person markers be suffixed to the stem", whereas in the imperfective prefixes are used to mark persons, and suffixes to mark gender and number [14]. The number features include singular (sing.), dual, and plural.

Table 1.4: Verbal Conjugation Example

Type	Person	Gender	Number	mood	Word	Gloss
Perfective	1st	M	Sing.	NA	katabtu	I wrote
	2nd	F	Sing.	NA	katabti	You wrote
	3rd	F	Plural	NA	katabna	they wrote
Imperfective	1st	F	Sing.	Ind.	taktubtu	She writes
	1st	F	Plural	Ene.	naktubw	we write
	2nd	M	Dual	sub.	taktubA	You write
	2nd	F	Sing.	Indi.	taktubyna	You write
	3rd	M	Plural	Jus.	yaktubw	they write

The imperfective verbs, which correspond to the English verbs in present tense, have four moods; namely indicative (ind.), energetic (eng.), subjunctive (sub.), and jussive (jus.), the indicative and the energetic moods require appending suffixes whereas the other two require removing suffixes.

1.2.3 Non-concatenative Morphology

There is also the "Broken Plural" (BP) category in nouns, which follows no conspicuous derivational procedure regarding the transformation from the singular to the plural form. It also demonstrates the non-concatenative nature of the Arabic morphology. This property can be fairly analogised to the English language irregular plurals as in "phenomenon → phenomena" or "datum → data", where the transformation from a singular to a plural form is not performed by appending the plural suffix "s" to the singular form, as the case with the regular plurals such as "house → houses". The operation requires rearranging the singular stem and adding in some cases new affixes to generate the plural form.

On the contrary to the sound plurals, which merely require, as Table 1.3 demonstrates, the appending of case-marking suffixes (An, At, wn, and yn) to the singular form to produce the plural. The broken plurals have nothing in common with their singular forms except the root's. The singular stem is broken into segments and rearranged, with long vowels sometime inserted between segments, to produce the plural form. On the contrary, for the regular plurals or sound plurals as they are referred to in Arabic Grammar, the suffix appends directly to the singular form as shown in Table 1.3.

Table 1.5: Sound or Regular Plurals vs. Broken Plurals

Singular	Pattern	Plural	Pattern	Type	Gloss
kAfir	CACiC	kAfirwn	CACiC		infidel–infidels
mujrim	muCCiC	mujrimwn	muCCiC	Regular	criminal–criminals
muqAtil	muCACiC	muqAtilwn	muCACiC		fighter–fighters
rajul	CaCuC	rijAl	CiCAC		man–men
bayt	CaCC	buywt	CuCwC	Broken	house–houses
masjid	maCCiC	masAjid	maCACiC		mosque–mosques

1.3 Arabic Morphology and IR

AIR has received more attention in research recently mainly for two reasons. The most obvious is the vast growth of the number of the Arabic web users which has exceeded 138 million users according to a July 2014 estimate [77]. Another significant reason is that the Arabic language was included as one of the tracks in the Text Retrieval Conference (TREC). That in turn resulted in the creation of a large Arabic corpus made available to researchers, which was the product of a collaboration effort between the National Institute of Standards and Technology (NIST) and the University of Pennsylvania [29; 59]. Therefore, the existence of a good Arabic retrieval system has become on demand, and tools available to develop it.

The majority of the existing research conducted so far is mainly divided into light stemming and root-based stemming [74]. The difference is that the light stemmer deals with inflectional morphology, and stops processing after removing a small set of prefixes and suffixes, whereas the root-based stemming tackles the derivational morphology as well, and so the processing continues until the root is found.

There have been several attempts to measure the AIR performance when a certain form of the word (surface, stem, or root) is chosen as an indexing term. Alkharashi conducted an experiment in the early 1990s on a relatively small corpus, which consisted of 355 documents in the area of Computer and Information Science. It also contained 1,126 keywords, 725 stems, and 526 roots [9]. The outcome of the study favoured the selection of the root as an indexing term, which proved its superiority to using the stem. Other later research undertaken by Aljlayl on the TREC corpus showed that using the stem improved the performance of their Arabic retrieval system and produced better results than the root [8]. The study utilised Khoja stemmer in finding the root [38].

The Buckwalter Arabic Morphological Analyzer (BAMA) was the first light stemmer used by many researchers before the TREC corpus existed [17]. It was developed by Tim Buckwalter after he left Xerox team, which built the Xerox Arabic Morphological Analyzer [12]. Several light stemmers emerged as a result of the existence of the TREC corpus though, with the light10 stemmer being the most well- known one followed by Al-Stem [20; 44].

In the root-based stemming category, two algorithms have surfaced in the past decade—Khoja and Sebawai. Khoja is a rule-based stemmer that strips off a number of prefixes and suffixes before matching the remaining part to a list of patterns and extract the root [38]. Sebawai on the other hand is based on a probabilistic model [20]. Darwish segmented 270,000 words from the TREC Arabic corpus, and 9,606 words from a classical Arabic collection called ZAD, using the Xerox Arabic Morphological Analyzer. His objective was to compile a list of word-root pairs, and then estimate the probability of the root based on its occurrence in the training data [12].

1.4 Research Questions and Goals

This study focuses on enhancing the Arabic stemming process by adopting a computational model that leverages the Quran as a morphological knowledge base. It attempts to draw the relationships between the query's terms and their morphological units, and in doing so, the root as well as the pattern of the keyword have to be known. It is the objective of this research to design a light stemming algorithm and a root-based stemming algorithm that would improve the performance of AIR. Through studying the Quran morphology, the relationships between prefixes, patterns, and suffixes are defined. We attempt to answer the following questions:

1. What is the most efficient indexing term, is it the stem, or the root?
2. What is the impact of limiting the legal prefix-pattern-suffix combinations to those allowed in the Arabic language, on the stemming process?
3. What are the minimal requirements for a good lexicon that can be utilised to enhance AIR performance?

The above questions are answered from the lexicon that was developed, and the following objectives are accomplished by this thesis:

- The development of a root-based and a light stemming algorithms based on the morphological rules inferred from the Quran.
- The comparison of the light stemming algorithm to light10, and Al-Stem using TREC-2002 Arabic corpus.
- The comparison of the root-based stemming algorithm to Khoja and Sebawai.
- The proposal of additional constraints on the stemming process to contain the over-stemming problems.

1.5 Organisation

Chapter 2 gives a brief introduction to the Arabic language. Section 2.1 goes over the Arabic orthography and how one letter could have multi-shape surface forms, whereas the Arabic phonology is shown in section 2.2. Those phonological rules that have direct effect on the morphological representation are detailed, because they present a challenge as far as the Arabic stemming goes. The Arabic morphology will be touched upon in section 2.3, the derivational morphology is demonstrated in section 2.3.1 by presenting the two derivational pillars; namely the roots in section 2.3.1.1, and the patterns in section 2.3.1.2. The inflectional morphology is exposed in section 2.3.2 by detailing the prefixes in section 2.3.2.1, and the suffixes in section 2.3.2.2. We will present the state-of-the-art Arabic stemming algorithms in Chapter 3. The root stemmers will be first introduced in section 3.2, whereas the light stemmers will be gone over in section 3.3. The Simple Arabic Stemmer (SAS) is presented in Chapter 4. The evaluation and the results of the experiments will be analysed in Chapter 5. In Chapter 6, we explain the research contribution to the Arabic Information Retrieval field in section 6.1, the limitations faced during the undertaking of the research are spelled out in section 6.2, we finally draw conclusions and give ideas for future research directions in section 6.3.

Chapter 2

Preliminaries

The Arabic language was mostly a spoken language before the Quran descended; only a handful of written poems had been known prior to Muhammad's (PBUH) time. The Seven Poems, or "The Hanging Poems", as they are referred to, constitute an exception [69]. They had been written, and hung on the Kaaba's wall in Mecca until the Quran was revealed, as one explanation for the name suggests.[1] They are well known and memorised by many Arab scholars, because they expose the Classical Language of that era, and delineate the clarity of the tongue those Arabs, to whom the Quran was talking, had in comparison to the 21st century Arabs.

The Quran acquired both credibility and admiration from the Arabs, even from those who had opposed its message, mainly due to its new style of phrasing, and how patterns and roots were used in generating words. These unique features made the Arabs both interested in listening to more of it, and baffled by its ability to form such sophisticated passages. The key characteristic of the Quran is that it presents a challenge that is centred in essence on "if you are in doubt of what we have revealed to our slave, write a single chapter such as this, and call upon all of your supporters, if you don't and you will not be able to, then you better believe in it in order to avoid hell fire" Quran $[2 : 23 - 24]$. This "very simple" challenge was introduced not only to the Arabs, but also to all people, and it is said that no one has answered the challenge.

[1] Others argue that the name originated from the fact that those poems are hung on the minds of the Arabs, and will never be erased from the Arabic literature.

The Quran had also provided the scholars with the Arabic morphological rules, which assisted beginners to methodically elevate their fluency. It became necessary for all Muslims to learn the Arabic Language so that they would grasp the messages embedded within the Quran verses.

A hundred years after the Quran was revealed, in the middle of the 8th century AD, the Arabic language studies gained formality, and schools were established for the purpose of teaching the Arabic language and its different fields including literature, morphology, and grammar [66]. The Arabic Grammar Schools are divided into two main branches; Koofah and Basrah, named after the cities where those scholars had gone to study, for example if they had studied in Koofah, then they follow the Koofah's principles [65]. These two schools differ on a number of issues such as the order of the alphabet letters, but they also have a unified view when it comes to the Language pillars such as the number of letters, the roots, and the patterns used in the derivation.

It is worth noting at this point that each example borrowed from the Quran would be accompanied by the Quran Index (QI). Three integers separated by dots represent the **chapter**, the **verse** in which the word is mentioned, and the **word**'s number. It is of the form: **chapter**.**verse**.**word**, so the QI 72.14.3 would represent the third word of the fourteenth verse of the seventy second chapter, and that is the word "Almuslimwna" (الْمُسلِمُونَ) "*the Muslims*".

In this chapter, the Arabic language is introduced briefly, including its orthography, phonology and morphology. In phonology the focus will be on the characteristics of the nominal weak stems; these are the phonological operations that influence the shape of the weak letters on the surface, and consequently affect the stemming process. The morphology section touches upon the Arabic lexicon and its major components: roots, patterns, prefixes, and suffixes.

2.1 Orthography

The orthography of a language is its spelling system, or the alphabet letters combined together to represent how the word is pronounced. As Tim Buckwalter once described the issues encountered by researchers when studying the Arabic morphology "The salient issues facing contemporary Arabic morphological analysis are summarised as predominantly orthographic in nature, although the issue of how to integrate morphological analysis of the dialects into the existing morphological analysis of Modern Standard Arabic is identified as the primary challenge of the next decade." [74].

So the orthography of the Arabic language itself poses a challenge due to the fact that there has not been a unified scheme for the Arabic digital scripts until the introduction of the unicode, before then, IBM, Apple, and Microsoft had their own different Arabic transliterations. There are also similar letters in the Arabic language that many Arabic writers find hard to distinguish from each other, the letter "h" (ـه) for example, is often written in the place of the letter "p" (ـة).

The Arabic alphabet consists of 28 main letters that are used in everyday language. These letters are written from right to left in a cursive style as Table 2.1 shows.

Table 2.1: The Arabic Alphabet Letters

Name	Unicode	Buck.	Unconn.	Connected with Preceding letter	Connected with Following letter	Connected with Both
ALIF WITH HAMZA	0623	>	أ	ـأ	أ	ـأ
BEH	0628	b	ب	ـب	بـ	ـبـ
TEH	062A	t	ت	ـت	تـ	ـتـ

Table 2.1: The Arabic Alphabet Letters

Name	Unicode	Buck.	Unconn.	Connected with Preceding letter	Connected with Following letter	Connected with Both
THEH	062B	v	ث	ـث	ثـ	ـثـ
JEEM	062C	j	ج	ـج	جـ	ـجـ
HAH	062D	H	ح	ـح	حـ	ـحـ
KHAH	062E	x	خ	ـخ	خـ	ـخـ
DAL	062F	d	د	ـد	د	ـد
THAL	0630	*	ذ	ـذ	ذ	ـذ
REH	0631	r	ر	ـر	ر	ـر
ZAIN	0632	z	ز	ـز	ز	ـز
SEEN	0633	s	س	ـس	سـ	ـسـ
SHEEN	0634	$	ش	ـش	شـ	ـشـ
SAD	0635	S	ص	ـص	صـ	ـصـ
DAD	0636	D	ض	ـض	ضـ	ـضـ
TAH	0637	T	ط	ـط	طـ	ـطـ
ZAH	0638	Z	ظ	ـظ	ظـ	ـظـ
AIN	0639	E	ع	ـع	عـ	ـعـ
GHAIN	063A	g	غ	ـغ	غـ	ـغـ
FEH	0641	f	ف	ـف	فـ	ـفـ
QAF	0642	q	ق	ـق	قـ	ـقـ
KAF	0643	k	ك	ـك	كـ	ـكـ
LAM	0644	l	ل	ـل	لـ	ـلـ
MEEM	0645	m	م	ـم	مـ	ـمـ
NOON	0646	n	ن	ـن	نـ	ـنـ
HEH	0647	h	ه	ـه	هـ	ـهـ
WAW	0648	w	و	ـو	و	ـو
YEH	064A	y	ي	ـي	يـ	ـيـ

It is obvious that some of these letters, the second letter “b” as an example, has four writing styles reflecting its position within the word, at the beginning, in the middle, or at the end, and one other style when being unconnected.

The Arabic alphabet letters are segmented into two groups; sound and weak.

The sound letters are 25, and they are called sound because they keep their shapes intact. The weak letters on the other hand are three letters, “>, w, y”, and are considered weak because they do not always keep their original shapes on the surface. And in some cases, certain phonological operation mandate the assimilation of the weak letter, and hence it is not shown on the surface.

Consider the two words in Figure 2.1 where the root is “wqt” (وقت) (“*having to do with time*”), whose first radical is the long vowel “w”. The two patterns differ in the type of the short vowel that is placed above the derivational prefix “m”, which also happens to precede the slot where the long vowel “w” should be inserted into, on the left there is an “i” while on the right there is an “a”. There

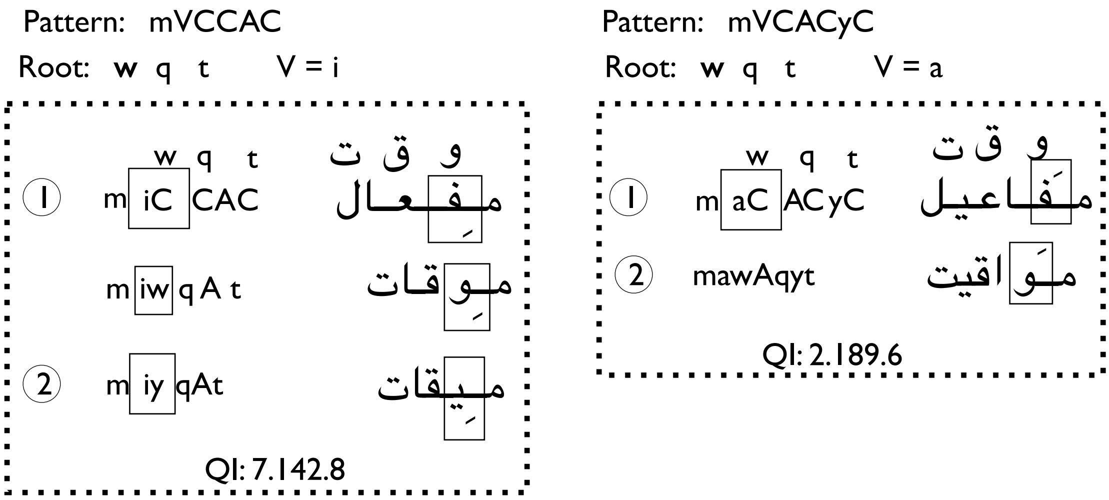


Figure 2.1: Long Vowels Phonological Nature Example.

are two steps undertaken to produce the stem “miyqAt”, meaning “*a specific time*”, the first is to place the root radicals and the short vowels into their slots in the pattern template. In the second, a special consideration has to be given to the long vowel slot because it is hard, if not awkward, in Arabic to pronounce an “i” followed immediately by the “w”, consequently the vowel “w” changes its shape on the surface to another vowel that has a similar sound to an “i”, in this case the “y”. The pattern on the right has an “a” right before the long vowel “w”, which leaves the vowel “w” intact on the surface, because the “w” can be pronounced with ease after an “a”. That is why these three letters are called weak; simply because they are not strong, or sound enough to always keep the same shape on the surface.

There are two extra letters that are common in the Arabic language as nominal suffixes, namely the singular feminine suffix "p" (ة), which can be used to denote two Broken Plural patterns, and the suffix "Y", which is a vowel, and can also be used with two Broken Plural patterns.

Table 2.2: Nominal Suffixes

Name	Unicode	Arabic Letter	Buck.
TEH MARBUTA	0629	ة	p
ALEF MAKSURA	0649	ى	Y

Table 2.2 shows these two suffixes, when they are connected with possessive pronoun suffixes for instance, "p" (ة) transforms into the third letter "t" (ت), while "Y" (ى) turns into "A" (ا) to accommodate for the morphological operation.

The diacritics group is composed of eight symbols written above the letter, except the third, and the sixth, which are written below the letter to indicate its sound as Table 2.3 shows. The second letter of the alphabet "b" (ب) is used to show the shapes of the diacritics, and where they are placed with relation to the letter itself.[2] The letter "b" in this example can bear three short vowels to signify its sound, an "a" as in "*bargain*", a "u" as in "*bull*", or an "i" as in "*Birmingham*". The no sound (SUKUN) symbol is represented in row 8.

The geminate symbol, which signifies the doubling of the letter, is shown in row 7. Rows 1, 2 and 3 are called "tanween", which are attached only to indefinite nouns, and have the same function as the short vowels while bearing an "N" sound at the end like "an", "un", and "in" respectively. It is a rule in Arabic Morphology that the "tanween" can be appended only to purely Arabic words, and can not be used with foreign words (e.g., "Amryky" "*American*").

[2]It is a convention in the Arabic Schools to use the second letter "b" (ب) when learning how to pronounce the diacritics.

Table 2.3: Diacritics on the Consonant “b”

Num.	Name	Unicode	Diacritic	Buck.
1	FATHATAN	064B	بًا	bF
2	DAMMATAN	064C	بٌ	bN
3	KASRATAN	064D	بٍ	bK
4	FATHA	064E	بَ	ba
5	DAMMA	064F	بُ	bu
6	KASRA	0650	بِ	bi
7	SHADDA	0651	بّ	b~
8	SUKUN	0652	بْ	bo

2.2 Phonology

Phonology is the study of the speech sounds used in a language [78]. Most if not all of the Arabic phonological rules involve vowels in some form or another. As he detailed in his PhD thesis, most of which is devoted to the verbal system, Brame stated that weak stems; those stems which contain a vowel as one of the root's radicals used in the generation, constitute most of the rules needed for the Arabic phonology [14]. The first finite-state attempts were done on the phonological rules due to their direct influence on the morphological presentation [37]. Kaplan and Kay work on the phonological rules inspired Koskenniemi to develop what was later known as two-level finite-state transducers.

We will focus on the nominal system and the phonological operations that dictate the transformation of the vowels into other shapes on the surface. Table 2.4 shows the shapes the three vowels bear on the surface.

Table 2.4: Vowels Surface Shapes

Name	Unicode	Arabic Script	Buck.
HAMZA	0621	ء	’
ALEF WITH MADDAH ABOVE	0622	آ	|
WAW WITH HAMZA ABOVE	0624	ؤ	&
ALEF WITH HAMZA BELOW	0625	إ	<
YEH WITH HAMZA ABOVE	0626	ئ	{
ALEF	0627	ا	A

The lengthening rule replaces two long “>, A” (ا، أ) vowels with a lengthy one as the second row of Table 2.4 shows, which is stressed by a long sound of the “A” letter (آ). Arabic phonologists represent it by pronouncing the long vowel “A” in a continuous sound while counting on the fingers to three, five, or seven. See Table 2.5 for an example from the lexicon.

Table 2.5: The Lengthening Rule

Pattern	Word	Ar. Word	QI	Gloss
CACiC	|xi*N	آخِذٌ	11.56.12	taker

This is Wright’s rule number 135, which states that “at the beginning of a word, if an ‘A’ (ا) production follows a radical ‘>’ (أ), the two ‘alifs’ are combined into one, which is written either as Maddah آ...” [81] (page 75).

Brame also had this rule as Rule II on page 30 of his thesis, where he replaces the two “a”s with a long “A” [14]. We have “>” (أ) as the initial radical of the root and the “A” (ا) vowel in the pattern template CACiC (فَاعِل), so >ACiC (

أأعِل) becomes |CiC (آعِل).

Another rule addresses the transformation of the middle voweled radical into the “}” (ـئـ) letter, as in Table 2.6 if preceded by an “A” and followed by the short vowel “i” to form the segment “ACi”. Wright’s rule number 240 on page 145 is the same as this one except the addition of the long vowel “>” here, his rule states “In the nominal agents of the first form of verb radical ‘w, y’ (و، ي), the place of the middle is occupied by ‘}’ (ئ) with hamza”.[3]

Table 2.6: Vowel-toYeh-With-Hamza-Above Transformation

Pattern	Word	Ar. Word	QI	Gloss
	AlbA}is	البَائِس	22.28.19	disperate
CACiC	daA}im	دَائِمٌ	13.35.11	permanent
	waDaA}iq	وَضَائِقٌ	11.12.7	distressed

The last rule is applied to all roots with a vowel as their last radical, when produced using patterns that end with the segment “AC”, as in the nominal pattern “CaCAC” (فَعَال).

Table 2.7: Vowel-to-Hamza Transformation

Pattern	Word	Ar. Word	QI	Gloss
CaCAC	barA’	بَرَاء	43.26.7	innocent
CuCAC	duEA’	دُعَاء	2.171.11	supplication
CaCAC	jazA’	جَزَاء	2.85.29	reward

Since there is an “A” (ا) right before the last radical in the pattern, the vowel

[3]See also rule number 133 on page 75, which is applied to this type of weak stems.

turns into a “HAMZA” (ء) on the surface (row 1 in Table 2.4). Table 2.7 shows the supporting examples from the lexicon, where the three vowels turn into “HAMZA”.

2.3 Morphology

Morphology is the study of the internal structure of words and how they are formed, and it is broken down into two subclasses: derivational and inflectional. The following are the definitions of the major components of Arabic Morphological lexicon:

- **morpheme:** a small unit of the lexicon that can be part of the word's structure (prefix, root, vowel, pattern, suffix).
- **root:** the only morpheme in the lexicon that can bear a meaning, consisting of three, sometimes four or five consonants, from which other words are derived.
- **pattern:** a combination of the root's radicals, and the short vowels, arranged according to a precise sequence, and it comes in a CV template.
- **radical:** the consonants of the **root** that was employed in the derivation of the word.[4]
- **affix:** a morpheme attached to the stem to be part of the surface word, an affix is of two types: a prefix or a suffix.

A word in the Arabic language can be a noun, a verb, or a particle. Nouns and verbs are created by derivational rules, which indicates that those words can be broken into roots and patterns. The particle category includes conjunctions, prepositions, or concrete words that are not derived from other words, and have no meaning by themselves unless accompanied by other words (e.g., “*for*” in English).

The first step in the formation of a word is the derivation stage, which leads to a new stem produced by intersecting the root and the pattern. The second

[4] Any letter of the alphabet, including the long vowels, can be a radical.

is inflectional, which involves adding other morphemes to the stem to form the surface word, but the meaning is left intact.

The roots and the patterns represent the two pillars of the Arabic derivational morphology. As for the inflectional morphology, there are prefixes and suffixes, which can be attached to stems to create surface words. They can be concatenated together to form other affixes, for instance a prefix could be one, two , three or four different prefixes combined into one. Infixes on the other hand place themselves in the inner part of the word [50]. Each category represents different features of the surface word. The following sections explain the function of each category, and its contribution to the word generation process.

2.3.1 Derivation

The derivational morphology is the process of generating a word with a new meaning by inserting the root's radicals into the pattern template. It sets the rules, and the constraints for inter-digitising the roots and the patterns, by permitting certain roots with a particular pattern, after all not all roots can be used with all patterns. The derivational morphology consists purely of roots and patterns, with these two morphemes, one can generate so much vocabulary with ease [70; 71].

2.3.1.1 Roots

The root in the Arabic language represents the cornerstone of the lexicon, on which so many derived words can be based. It is the unit that determines the meaning of the word; the patterns merely give the templates from which different related words can be produced, but the meaning is in the root. Many morphologists have studied the roots, but there was an 8^{th} century scholar who even went so far as to find the relationship between the different combinations of the radicals for a particular root. His name is Alkhalil (الخَليل), and is considered by many scholars of the language to be the master of the Arabic Morphology.

In his book "AlEyn" (العين), he ordered the alphabet letters according to their sound, and how far from the lips and towards the throat their pronunciations require the sound to originate from [64].

He named his book after the 18th letter of the alphabet, "E" (ع), this letter happens to be the deepest in the throat in terms of where the sound is coming from. Wright describes this letter by saying "The correct pronunciation of some of these letters, for example 'E' (ع), it is scarcely possible for a European to acquire, except by long intercourse with natives" [81].

The order of the alphabet letters is according to the source where the sound of that letter is initiated from in the throat, starting with the "E" (ع) being the furtherest, and ending with the "m", which demands no more than closing the lips slightly while making the sound. In Alkhalil's lexicon, the relationship between the root's radicals are defined, and it is not long before one can conclude that the root ,or as he referred to it, the "Substance" (المادَّة), is the unit from which the meaning is recognised.

To make this point clear, consider the three radicals "q w l" (ق و ل) in Table 2.8 with a meaning related to "movement and light-weight".[5]

Table 2.8: The Root "qwl" (قول) and its Combinations

Root	Meaning	Ar. Meaning	Ar. Root
qwl	saying	القول: الفمُ وَاللِسَانُ يَخِفَّانِ لَه	قول
qlw	strong donkey	القِلو: الجَحشُ الفَتِيُّ الذي يُركَب	قلو
wql	mountain horse	فَرَسٌ وَقِل: حَسَنُ الدُّخُولِ بينَ الجِبَال	وقل
wlq	move faster	ولقَ يلقُ إِذَا أَسرَع	ولق
lwq	stirring food	مَا آكل من الطَّعَامِ إِلَّا مَا لُوقَ لِي	لوق
lqw	eagle	لَقوة: العُقَابُ السَّريعةُ السَّير	لقو

[5] According to our lexicon, "qwl" (قول) is the most used root in the Quran.

Table 2.8 shows that all of the six combinations of the three radicals bear a meaning related to either, being in motion, or light, and that is how Alkhalil organised his lexicon.

Not so many Arabs have studied his work mainly due to its complexity, and it requires a deep knowledge of the pure Classical Arabic, namely spoken Bedouin Language of that time. Alkhalil verified most of his dictionary items by asking the Bedouins where the root originated from, and how they used it. The linguists who came after him were definitely influenced by his work with no exception [70].

Alkhalil also set the principles for the derivational morphology of the language by limiting roots to three, four, or five radicals, and any more letter in the word is supplementary. The three-radical, or triliteral roots according to another more comprehensive lexicon "LisAnu AlEarab" "*The Arabs Tongue*", constitute the majority; they reach 6,538 roots, compared to 2,548 four-radical, or quadriliteral roots, and 187 five-radical or pentaliteral roots [63].

2.3.1.2 Patterns

Patterns in the Arabic morphology represent the containers, into which roots and short vowels are moulded to form the word. The verbal patterns are very small in number, 15 patterns, 12 of which are of common use. Table 2.9 shows the first and the most popular triliteral verbal pattern in Arabic [81]. According to Arabic morphologists, the verbs are rooted into the 3rd person.singular.masculine.perfective forms, some even suggest that the root originates from this form, which is the same as the past tense form in the English language when referring to a masculine subject.

Table 2.9: Triliteral Verbal Pattern (Form I)

Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
CaCaCa	kataba	he wrote	كَتَبَ	فَعَلَ	6.54.9

For the quadriliteral verbs, there are three patterns, Table 2.10 shows a very common quadriliteral pattern in the Arabic language.

Table 2.10: Quadriliteral Verbal Pattern Example

Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
ACCaCC~a	ATma>nantum	felt ease	اطمَأَنَنْتُمْ	افعَلَلّ	4.103.11

Table 2.11 shows a sample of the nominal Active Participle (AP), which is used very often because it represents the verb as well as the subject. The pattern in Table 2.11 is the active participle of the first verbal pattern shown in Table 2.9. The noun "kAtib" "*writer*" conveys the identity of the subject, in addition to the type of action performed "*writing*", and that is why Classical Arabic speakers would rather use this type of noun more frequently. In different words, a single word can bear two implicit features.

Table 2.11: Triliteral Active Participle (Form I)

Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
CACiC	kAtib	writer	كَاتِبٌ	فَاعِل	2.282.14

Table 2.12 shows the active participle for the quadriliteral verbal pattern in Table 2.10.

Table 2.12: Active Participles for the Quadriliteral Verbal Pattern

Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
muCCaCiC~	muTma<n~	at peace	مُطْمَئِنٌّ	مُفْعَلِلّ	16.106.11

It is apparent that all active participles, including quadriliteral patterns, end with the segment “CiC”, except the 9th pattern “muCCaC~” (مُفعَلّ), which ends with “CaC”.

The passive participles play the same role as the active participles, except that it reveals the object rather than the subject. The first verbal form has the passive participle as maCCwC, so for the root “ktb”, “maktwb” “*written*” is the stem as in Table 2.13.

Table 2.13: Form I Passive Participle

Pattern	Stem	Gloss	Word	Ar. Pat.	QI
maCCwC	maktwbA	written	مَكتُوبًا	مَفعُول	7.157.8

The rest of the verbal forms, the 9th is the exception again, bears the same pattern as the active participles, with the last short vowel changed to an “a” instead of an “i” as in Table 2.14. In other words, the passive participle ends with the segment “CaC” instead of “CiC”.

Table 2.14: Form II Passive Participle

Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
muCaCCaC	musal~amap	handed to	مُسَلَّمَةٌ	مُفَعَّل	4.92.17

The next category is the infinitive, which according to Wright’s definition “expresses the action, passion, or state indicated by the corresponding verb, without any reference to object, subject, or time” [81]. Table 2.15 shows the infinitive form of the triliteral verbal pattern in Table 2.9.

Table 2.15: Infinitive for the Triliteral Verbal Form I

Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
CiCACap	tijArap	commerce	تِجَارَةً	فِعَالَة	4.29.13

Table 2.16 shows the quadrilateral infinitive form of the verbal pattern in Table 2.10.[6]

Table 2.16: Infinitives for the Quadrilateral Verbal Form

Pattern	Stem	Gloss	Ar. Word	Ar. Pat.
AiCCiCCAC	AiTmi<nAn	relief	اطمِئنَان	افعِلَّال

The nominal system consists of many adjectives, Table 2.17 shows one example of what is called the verbal adjectives, which usually indicate the state of the subject with a verb of some sort .

The pattern in Table 2.17 is used more frequently than any other pattern. For example, when it is combined with the root "mwt", the word "may˜it" "*dead*" is produced. Note how the vowel "w" is transformed into another vowel, namely the "y" due to the presence of the vowel "y" in the pattern itself, the same case is demonstrated in Figure 2.1.

Table 2.17: Verbal Adjectives

Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
CayCiC	may˜it	dead	مَيِّتٌ	فَيعِل	39.30.2

The pattern in Table 2.18 is the intensive adjective, which indicates, according to Wright "a very high degree of the quality which their subject possess".

Table 2.18: Intensive Adjectives

Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
miCCyC	miskynA	poor	مِسكِينًا	مِفعِيل	90.16.2

The nouns denoting instruments are frequently used in Arabic, table 2.19

[6]This is the same pattern as Pattern IV in item 203 on page 117 of Wright's Grammar Book.

shows a sample pattern, it can produce so many useful words, one such word is "qAnwn" (قَانون) "*law*", which has the same pattern, but with the root "qnn" (قنن).

Table 2.19: Triliteral Noun of Instrument

Root	Pattern	Stem	Arabic	Gloss	QI
nqr	CACwC	AlnAqwr	النَّاقُورِ	trumpet	74.8.4

The other type of plurals in Arabic are called Broken Plurals (BPs), because deriving the plural form from the singular is not a straightforward concatenative operation where plural suffixes are appended to the singular form.

Table 2.20: A Popular Triliteral Broken Plural Pattern

Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
CuCACY	sukArY	drunks	سُكَارَى	فُعَالَى	4.43.9

In her PhD thesis, Levy addressed the sound plurals and stated that their rules can be applied to the derived words, or words that are formed from other words by rules of morphological derivation (e.g., participles) [45]. McCarthy has also analysed this very thoroughly, when he applied his prosodical theory on this type of Arabic plurals [51].

Neme also studied the transition from the singular to the broken plural through the traditional description that "The path from a singular form to a BP passes through a root." [55]. Table 2.20 shows a popular pattern in Arabic Broken Plurals, it also demonstrates the usage of the letter "Y" of Table 2.2, which was stated that it can be used to denote two broken plural patterns.

We will touch upon this topic again in Chapter 5 when we analyse the lexicon. Table 2.21 shows a quadriliteral Broken Plural Form, which is very common in

Arabic. For example, it is the plural of the word "Hunjurap" (حُنجُرَة) "*throat*".

Table 2.21: A Popular Quadriliteral Broken Plural Pattern

Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
CaCACiC	AlHanAjir	throats	الحَنَاجِر	فَعَالِل	33.10.13

2.3.2 Inflection

The inflectional morphology describes predictable changes a word undergoes as a result of syntax [41]. Its operations are always performed after the derivational operations, just like in English, making the stem "writer" plural, to become "writers" would happen after we derive it from the root "write". In the next two sections, we present the two main types of the Arabic inflectional morphology—the prefixes, and suffixes.

2.3.2.1 Prefixes

The prefixes are very important in the word structure, because besides the change that the word has to undergo to reflect their presence under some conditions (e.g., first letter is a vowel), they bear grammatical features. Nominal prefixes in Arabic are divided into four groups; interrogation, conjunction, preposition and the definiteness marker "Al" meaning "*the*".

All of the prefixes are composed of a single letter except the definiteness marker "Al", which has two letters as Table 2.22 demonstrates. The sequence in Table 2.22 above reflects the order for Arabic prefixes, for instance a combination can have an interrogative particle followed by a preposition particle, or a preposition particle preceding the definiteness marker "Al".

Table 2.22: Nominal Prefixes Categories

Interrogation	Conjunction	Preposition	Definiteness Marker
> (أ)	w, f (و، ف)	b, k, l (ب، ك، ل)	Al (ال)

When studying the different combinations of prefixes in Arabic, general observations can be made:

1. No prefix can be added after the definiteness marker "Al", and only the interrogation particle can precede the conjunction particles "w, f".

2. Preposition prefixes always come after the conjunction "w, f" and before the definiteness maker "Al".

3. The preposition particles "b" and "k" must have the suffixes "tyn, yn" to denote the genitive case for the dual and plural cases.

4. The "l" prefix is included in the previous rule, also when used with the definiteness marker "Al", the letter "A" is assimilated, to become "ll".

Nouns with no prefixes constituted more than half of our lexicon, the definiteness determiner "Al" is next followed by the conjunction particle "w", and then the preposition particle "b".

The verbal prefixes are used only with the imperfective verbs to mark the person as Table 2.23 demonstrates.[7] In contrary to the perfective form, which (as we will see) uses only suffixes to denote the person with no prefixes at all. For the imperfective verbs, there are certain suffixes that must accompany the prefixes as indicated by the its colour in Table 2.23.

[7]The content is copied as is from of Brame's PhD thesis (item 3 on page 6).

Table 2.23: Imperfective Conjugation Prefixes

Person	Gender	Singular	Plural	Dual
1st		>ktub "*I write*"	naktub	
2nd	m.	taktub "*you write*"	taktubw	taktubA
	f.	taktuby "*you write*"	taktubna	taktubna
3rd	m.	yaktub "*he writes*"	yaktubw	yaktubA
	f.	taktub "*she writes*"	yaktubna	taktubA

2.3.2.2 Suffixes

There are two types of suffixes that can be concatenated to the Arabic nominal patterns. The first is inflectional and used to signify the number, the gender, and the case of the noun. Table 1.3 shows the suffixes that are used to indicate the nominal number, gender, and case endings. The second type is what is known as the possessive pronouns, which can be represented in Arabic with eleven suffixes. Two of these are for first person, four are used for 2nd person and 5 for 3rd person. For example, if we have the word "bayt" (بَيت) meaning in Arabic a "*house*", then "bayty" would be "*my house*" and "baytnA" is "*our house*" and so on. Table 2.24 shows the possessive pronouns suffixes, which are the same as the verbal object pronouns.

Table 2.24: Possessive Pronouns

Person	Gender	Singular	Plural	Dual
1st		kitAby "*my book*"	kitAbunA "*our ...*"	
2nd	m.	kitAbuka "*your ...*"	kitAbukum	kitAbukumA
	f.	kitAbuki	kitAbukun~a	kitAbukumA
3rd	m.	kitAbuh "*his ...*"	kitAbuhum	kitAbuhumA
	f.	kitAbuhA "*her ...*"	kitAbuhun	kitAbuhumA

The verbal suffixes are used with the perfective forms to denote the person, and with the imperfective form to accompany the prefixes that indicate the person. Table 2.25 shows the perfective conjugation affixes, which includes the prefixes and the suffixes that must accompany them.[8]

Table 2.25: Perfective Conjugation

Person	Gender	Singular	Plural	Dual
1st		katabtu "*I wrote*"	katabnA	
2nd	m.	katabta "*you wrote*"	katabtum	katabtumA
	f.	katabti "*you wrote*"	katabtunna	katabtumA
3rd	m.	kataba "*he wrote*"	katabw	katabA
	f.	katabat "*she wrote*"	katabna	katabatA

2.4 Conclusion

We have shown certain Arabic morphological aspects that demonstrate the complexity of the Arabic stemming as a computational task. In section 2.2, we have gone over a few of the phonological features that have a direct impact on the stemming process. Furthermore, we have drawn the relationship between the roots and the patterns and how they are used in the Arabic language to produce

[8]The content is copied as is from Brame's PhD thesis (item 2 on page 5).

new words. We will also detail the relationships between different patterns in chapter 4, and give a more penetrating analysis of how certain nominal patterns are related to others (i.e., singular to plural).

The next chapter talks about the state-of-the-art root and light stemmers, and previous AIR experiments. There is always that guilt of neglecting, or omitting something, and it is true in this case, because the Arabic language, as the popular saying goes "is an ocean without a shore" [72]. Therefore, this attempt in tackling the Arabic computational morphology is by no means inclusive of all elements in the Arabic lexicon, however the patterns and the roots we present in this thesis are very common, and represent major components of our stemmer lexicon, which is detailed in section 4.4.

Chapter 3

Current Stemmers

There have been several Arabic Information Retrieval (AIR) experiments over the last decade, but AIR research had its roots back in 1990 in the United States. That resulted in an advancement in the field, though when compared to other languages such as English, the progress has been slow. The complexity of the morphology is the main reason why the Arabic language still lacks of NLP tools capable of generating the correct morphological analysis for all of the language vocabulary.

We will present the different attempts of tackling the Arabic stemming issues in AIR, and the approach adopted by each stemmer. There are generally three types of stemmers: manually-constructed lexicon, which requires manual effort to build and maintain, light stemming, which strips off a small set of affixes to reach the stem, and root stemming, which removes the affixes, and then matches the stem against a predetermined list of patterns to extract the root. As we will learn, each step depends on the previous one; the root stemming task requires an error-free stem from the light stemming module, and also the root has to be reconstructed to rectify weak radicals if any phonological operations took place during the derivation process.

The Modern Standard Arabic language is characterised by the absence of diacritics, short vowels in particular, which adds another challenge to the stemming operation, and sometimes leads to more than one solution. This ambiguity is usually resolved only by human intervention, someone with a background in morphology, who can select the analysis that best fits the input word. Consider the example in Table 3.1, the two words ("faraAga") are identical on the surface, while they differ greatly when another deeper look is taken.

Table 3.1: Ambiguity with Full Vocalization

Pattern	Root	Vowel	Prefix	Suffix	Word	Gloss	Arabic
CVCVC	rwg	a	fa	a	faraAga	swerved	فَرَاغَ
CVCVAC	frq	a	∅	a	faraAga	emptiness	فَرَاغَ

The first word is a verb with the pattern Form I "CVCVC", the weak root "rwg", the short vowel "a", and the conjunction particle "fa", with a meaning close to "*then*" as a prefix. An example is "faraAga <lY >hlhi" (فَرَاغَ إِلَى أَهلِهِ) "*then he went unnoticed to his family*". It is worth pointing out that the reason for the ambiguity is the weak root; because of the preceding short vowel "a" in the pattern template, the long vowel "w" changes its shape to "A". Had it not been the vowel transformation into an "A", the analysis would be as straightforward as finding the corresponding pattern.

The second word is the Individuality nominal pattern "CVCVAC", the root "frg", and the suffix "a" marking the accusative case (i.e., the word being an object of a verb). An example of usage is "taraktu faraga Al<jAbapi" "*I left the blank of the answer*" (تَرَكتُ فَرَاغَ الإِجَابَةِ).

This example demonstrates plainly the ambiguity usually faced by Arabic stemmers. Note that every radical is followed by a short vowel, in other words, the word is fully vocalised, and every letter has its own short vowel when the two suffixes are included. And yet, that is not informative enough to determine definitely the right meaning. Even a human judge would have hard time producing the correct analysis, unless some sort of semantic scrutiny is performed on the sentence that contains the word.

The root here depends on the chosen solution—the nominal pattern would result in a sound root, whereas the verbal analysis would produce the root “rAg”, and after another level of processing, the vowel “A” cannot be a root’s radical, therefore the root is rectified to become “rwg” (روغ).

As the following sections reveal, no stemming solution has surfaced that can produce a single analysis for the example given above, simply due to the lack of an advanced Arabic Semantical Analyser. The approaches studied here shed light on certain aspects of the complexity of Arabic morphology. We try to adopt a middle approach, which involves an automated stemming process with constraints to contain the over-stemming issue, and we use the Quran phonological rules to guide the root extraction process.

The Quran morphology provides the answers as to how the Arabic words are structured. We will utilise that knowledge to improve the stemming issue in AIR by adding more nominal patterns to cover a wider range of stem classes, restricting the removal of prefixes and suffixes to the legal combinations found in the Quran, and minimising the size of the lexicon.

3.1 Overview

There have been several attempts to find the root of an Arabic word–some of the work built on previous ideas [1; 3; 67; 75]. Moreover, there are several well-known Arabic stemmers made available for the scientific community, one of which is the Xerox Arabic Morphological Analyser, which is a commercial bundle of software built on finite-state machines. However, the Buckwalter Arabic Morphological Analyser (BAMA), Khoja, and Sebawai on the other hand, are freely distributed under a GPL Agreement, and can be downloaded from the Internet [17; 19; 38].[1]

Table 3.2 shows the features and limitations of publicly available stemmers.

Table 3.2: Current Arabic Stemmers Features and Limitations

Stemmer	Feature	Limitation
BAMA [17]	• Innovative Approach • Used in AIR research • Manually verified.	• Manually maintained • Multi-affix problem • No root or pattern
AMIRA [24]	• Uses Machine Learning • Tokeniser, POS, Analyser • Used by many institutes	• Limited to training • Missing patterns • No root or pattern
MADA+TOKAN [33]	• Based on BAMA • Uses a selection process • Adheres to Arabic	• Manually maintained • Missing patterns • No root or pattern
light10 [43]	• Limited affixes • Good in IR • No Arabic is needed	• Removes affixes blindly • No verbal affixes • No root or pattern
Al-Stem [20]	• Statistics-based • Simple to develop • No Arabic is needed	• Removes affixes blindly • Doesn't follow rules • Limited Affixes
Khoja [38]	• Rule-based • Produces the root. • Utilised by UMASS and IIT	• Removes affixes blindly • Limited patterns • Multi-suffix problem
Sebawai [19]	• Statistics-based • Produces the root • No Arabic is needed	• Huge root lexicon • Limited to training • Does not follow rules
XEROX [12]	• Built on Finite-State Machines • According to Arabic rules • Vowels processing is superb	• Commercial • Multiple solutions • Ill-formed words

Table 3.3 shows a numerical comparison of the lexical contents for each stem-

[1] http://zeus.cs.pacificu.edu/shereen/research.htm.

mer. The legal combinations refer to the allowed prefix-stem-suffix entries for BAMA, and the prefix-pattern-suffix entries for SAS. In this chapter, we will go over all the mentioned stemmers, and explain how they process an Arabic word in order to extract the stem, and the root.

Table 3.3: Current Arabic Stemmers Lexical Comparison

Name	root	Pattern	Prefix	Suffix	Legal Combination
BAMA	0	0	421	1,170	4,603
Light10	0	0	6	10	0
Al-Stem	0	0	25	21	0
Xerox	4,930	400	NA	NA	NA
Khoja	4,748	46	16	28	0
Sebawai	10,405	245	215	290	0
SAS	2,546	111	86	193	5,617

3.2 Root Stemmers

The root stemmer determines both the derivational and inflectional steps involved in creating the surface word. It strips the inflectional affixes off, and then matches the stem to a predetermined list of patterns in order to extract the root. This method obviously has its own challenges, one of which is the over-stemming problem, where a root's radical is deemed an affix and removed. It is also imperative to reach this stage of processing with an error-free stem in order to produce the correct root.

3.2.1 Xerox Arabic Morphological Analyzer

In 1989, Kenneth Beesley, Tim Buckwalter, and Stuart Newton described a working computer program that performs a morphological analysis and dictionary lookup of written, on-line Arabic words [12]. It started as an ALPNET project, and later was acquired by Xerox, and made commercially available. Although the core lexicon (prefixes, roots, vowels, patterns, suffixes) was essentially the same, Beesley had rewritten the finite-state rules for the Xerox system [10]. This was the first Arabic root stemmer, and was built using finite-state automata implementing what is known as the “two-level morphology”.

The concept of two-level morphology was initially introduced by Kimmo Koskenniemi in 1983, when he built a computational linguistic model for the Finnish language that was capable of word recognition, and production [40]. The rules were processed in a parallel fashion, instead of the sequential execution, which made the old generative phonology framework less applicable to complex morphologies. Koskenniemi had visited the United States in the early 1980s, and was exposed to the idea of two-level finite-state machines by Kaplan and Kay [37].

Koskenniemi's parallel model allowed for a bidirectional capability, whereas the old framework permitted only the production, and was considered unidirectional, meaning it cannot perform the analysis task. The idea was to have two levels of representation, lexical and surface, that are executed in parallel. This required building finite-state transducers (a regular expression that accepts an alphabet of paired symbols), and at that time, there was no tool capable of performing such a process automatically, so Koskenniemi had to hand-encode the rules into finite-state machines in order to implement his idea.

Koskenniemi empirically showed that languages with complex morphologies such as the Finnish language can be computationally represented as regular languages with the mathematical power of finite-state machines. That opened the door for attempts to apply such a concept to other languages, and encouraged other researchers to follow his footsteps.

Although there was about 6-year gap between the two systems, the Xerox system was built using exactly Koskenniemi's idea, but applied to Arabic. The speciality of the Arabic language dictated an extension to the original model, in order to account for the interdigitation of roots and patterns, or a ”Detouring” mechanism as Beesley called it [11]. The system consists of four finite-state transducers that are compiled in parallel to generate all words possible mainly by two formal operations: concatenation and subtraction.

Consider Figure 3.1, where the top tier represents the **filtering** automata, which is actually executed in parallel with the core lexicon automata. The **core**

lexicon is composed of 4,930 roots, and 400 patterns. The **intersect** rules facilitate the interdigitization of roots and patterns to form stems; a stem in the Xerox system is a combination of three lexical morphemes; the root, the vowels, and the pattern.

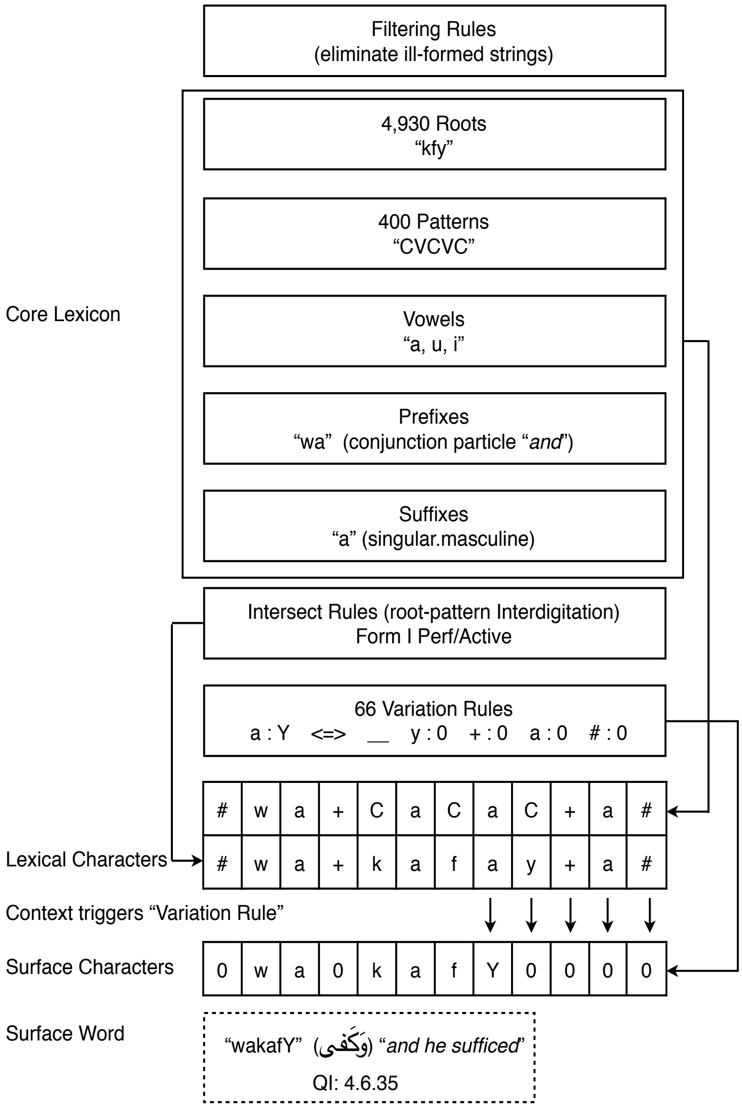


Figure 3.1: Xerox Analyser Step-by-Step Example.

There are 4,930 roots, each one of which is hand-encoded to specify the patterns with which it legally combines. Furthermore, each root entry with a nominal pattern stores more information pertaining to the twenty inflectional categories of noun suffixations, specifying what and how many suffixes are permitted, with an average of 3.54 inflectional suffixation categories per noun pattern morpheme, and 18 pattern morphemes per root [10; 16]. The system generated 90,000 stems, but when concatenated with all different combinations of affixes (prefixes and suffixes), 72,000,000 forms were produced. This is one of the downsides of the system-the over-generation problem-which is expected when concatenating all morphemes in the lexicon. Consequently, a filtering mechanism was mandatory to eliminate ill-formed strings.

It is the job of the filtering automata to handle the discontiguous dependencies between multiple morphemes in a single word. A rule that handles which prefixes are permitted to be concatenated with the nominal suffixes was required. For example, the preposition particle "b" requires the noun to be in a genitive state, therefore the singular "i", the dual "yn", and the plural "yn" represent the suffixes that can be attached to nouns in the genitive state. The plus "+", and the hash "#" signs are used to mark the morpheme, and the word boundaries respectively, and they are realised as 0, or empty on the surface.

In Figure 3.1, the lexical input contains the string "#wa+kafay+a#", however this string can not be realised on the surface as it is, because of the phonological nature of the root's last radical, being a vowel. The **variation** rules perform the mapping from the lexical to the surface representation. Sixty six finite-state rules were needed to handle the special cases of Arabic phonology and orthography of the form:

$$\mathrm{a} : \mathrm{Y} \Leftrightarrow \_y : 0+ : 0a : 0\# : 0$$

The rule shows the two-level concept by defining the relationship between the lexical and the surface characters. It states that a lexical "a" is to be realised as "Y" ("alif maqsurah") on the surface if and only if (⇔) it is followed by a lexical "y" realised as empty, a lexical morpheme boundary realised as empty, a lexical "a" realised as empty, and the end of the word.

This rule would be triggered, as Figure 3.1 depicts, when the weak root "kfy" ("related to sufficiency") is used with Form I perfective. Since the last radical is the vowel "y", and it is preceded in the pattern template by the short vowel "a", and followed by the suffix "a", the radical "y" has to be turned into a "Y" on the surface to reflect the proper sound. One of the characteristics of the Xerox system is that it was built in accordance with the the Classical Arabic rules, flexible enough to process semi-voweled or un-voweled words.

3.2.2 Khoja

Shereen Khoja developed a stemming solution that was later used by several researchers, including the UMASS, and IIT experiments [38]. The rule-based stemmer contains 3,822 three-radical (triliteral), and 926 four-radical (quadrilateral) roots, besides nine other lists of special roots, which are the roots that have a vowel, or a duplicate letter as one of their radicals.

The stemmer starts by checking the length of the word. If the word is a two-letter word then it moves to the two-letter function, which has a list of 34 two-letter words with the second radical duplicated to form a correct triliteral root. For instance, the entry "jd" would generate the root "jdd", where the second letter is just doubled to produce the third.

The stemmer then checks if the root is not found, it then checks a list of 359 two-letter words whose last radical is a vowel. Khoja had generated these lists manually by storing only two radicals of any geminate or weak root. For instance, if the first radical of a weak root was "w" as in "wqt" (وقت), then the author would store the root as "qt" (قت) in a list called "firstWaw".

Algorithm 3.1: Khoja "stemWord" Function

Input: a UTF Arabic word with length >= 2.

Algorithm 3.1: Khoja "stemWord" Function – continued

```
Output:  A root of 3 or 4 radicals, or the word if a root is not found.
function Stem(word)
    if wordLength = 2 then
        word ← isTwoLetter(word)
    if wordLength = 3 then
        word ← isThreeLetters(word)
    if wordLength = 4 then
        word ← isFourLetters(word)
    if rootFound = false then
        word ← checkPatterns(word)
    if rootFound = false then
        word ← checkDefiniteArticle(word)
    if rootFound = false and stopWordFound = false then
        word ← checkPrefixWaw(word)
    if rootFound = false and stopWordFound = false then
        word ← checkForSuffixes(word)
    if rootFound = false and stopWordFound = false then
        word ← checkForPrefixes(word)
    return word
end function
```

After checking whether the last letter is a vowel, the stemmer checks if the first letter is a vowel in a list of 215 entries, and if no root is produced, it checks the "middleWeak" list, which contains 525 items. If the word is a three-letter word, then the stemmer checks if any of the letters is weak, and attempts to rectify it.

The stemmer checks the 926 quadriliteral roots if a four-letter word is encountered. Algorithm 3.1 shows the function "stem", along with two main functions that are relevant. One is called "checkForPatterns" shown in Algorithm 3.2, which attempts to match the input word to a list of forty six patterns, and extract the root's radicals from the positions in the pattern where the letters "f, E, l" (ل ،ع ،ف) are present.

Algorithm 3.2: Khoja "checkPatterns" Function

Input: a UTF Arabic word with length >= 3 , and a list of 46 patterns.

Output: A root of length 3 or 4 radicals if the word matches any pattern.

```
function checkpatterns(word)
    word ← ∅
    modifiedWord ← ∅
    for pat ∈ patterns do
        if patLen = wordLen then
            numberSameLetters ← 0
            for i = 0 to wordLen
                ▷ Unicode for ("f", "E", "l") is (u0641, 0639, 0644)
                if pat[i] = word[i] and pat[i] ≠ u0641 and pat[i] ≠ u0639 and pat[i]
≠ u0644 then
                    numberSameLetters ← numberSameLetters + 1
            if wordLen − 3 ≤ numberSameLetters then
                for i = 0 to wordLen
                    if pat[i] = u0641 or pat[i] = u0639 or pat[i] = u0644 then
                        append(root, word[i])
                modifiedWord ← root
                modifiedWord ← isThreeLetters(modifiedWord)
                If rootFound then
                    word ← modifiedWord
                    return word
    return word
end function
```

If the root is not found yet, the "checkPattern" procedure matches the word to a list of 46 patterns, and if successful, it returns the root after verifying it. No affixes have been removed up to this point, so the stemmer checks for the definite article and its derivatives (wAl, bAl, kAl, and fAl). If none exists, then the stemmer moves to the next step in the program. Checking for the prefix "w" ("*and*") is performed right after checking for the definite article, and if no root is found, then the stemmer checks for suffixes and prefixes.

Algorithm 3.3: Khoja "checkForPrefixes" Function

```
Input:  A word of length ≥ 3, a list of 16 prefixes.
Output:  A root of length 3 or 4 radicals after removal of any matching prefix.
function checkForPrefixes(word)
        modifiedWord ← word
        for prefix ∈ prefixes do
            if prefix ∈ modifiedWord
                modifiedWord ← substr(modifiedWord,0,prefixLen)
               if checkStopWord(modifiedWord) then
                    return modifiedWord
               if modifiedWordLength = 2 then
                    modifiedWord ← isTwoLetters(modifiedWord)
               else if modifiedWordLength = 3 then
                    modifiedWord ← isThreeLetters(modifiedWord)
               else if modifiedWordLength = 4 then
                    modifiedWord ← isFourLetters(modifiedWord)
                if not rootFound and modifiedWordLen > 2
                    word ← checkPatterns(word)
                if not rootFound and not stopWordFound and not fromSuffixes then
                    modifiedWord ← checkForSuffixes(modifiedWord)
                if stopWordFound then
                    return modifiedWord
                if rootFound and not stopWordFound
                    return modifiedWord
        return word
end function
```

The function "checkForPrefixes" is invoked as a last step in the "stemWord" procedure to check the start of the word against a list of 16 prefixes in the following order; "Al,wAl,bAl,kAl,fAl,ll,l,A,w,s,b,y,n,m,t,f". Algorithm 3.3 shows that when a prefix is stripped off, the remaining stem is checked if it is a stop word.

Next, the function “checkPatterns” is called again to find out if the word matches any of the 46 patterns in Khoja’s lexicon, given that the word is not a stop word and its length is 2. When a root is not reached yet, the function makes sure that the remaining word is not a suffix, if not then Khoja checks to see if the word contains any of the 28 suffixes in the lexicon. Finally, if no root has been found yet, then “checkForPrefixes” returns the reaming word to the “stemWord” procedure.

3.2.3 Sebawai

Kareem Darwish at the University of Maryland developed a light stemmer (Al-Stem) and a morphological analyser (Sebawai) based on the statistics of a corpus he compiled [20]. The idea evolves around building a shallow morphological analyser without utilising a lexicon or a rule-based engine.

He leveraged the Xerox Arabic morphological analyser to compile a list of word-root pairs from a 14th century classical book called Zad AlmaEAd. The morphological analyser was successful in finding the roots of 9,606 words. Another list was constructed from the Linguistic Data Consortium (LDC) Arabic corpus containing AFP newswire stories [30].

Algorithm 3.4: Sebawai “gen_stems” Function

Input: A word of length ≥ 3.
Output: A root of 3 or 4 radicals,
function gen_stems()
    temp = prefix_str = suffix_str ← ∅
    for i = 0 **to** word_length **do**
        for j = 0 **to** word_length **do**
            temp ← word

Algorithm 3.4: Sebawai "gen_stems" Function – continued

```
if substr(word,word_length-j,j) ∈ ssuf then
    suffix_str ← substr(word,word_length-j,j)
    suffix ← ssuf[sufix_str]
else
    suffix ← −1
    suffix_str ← #
if substr(word,0,i) ∈ spre then
    prefix_str ← substr(word,0,i)
    prefix ← spre[prefix_str]
    subsr(temp,0,i) ← ∅
else
    prefix ← −1
    prefix_str ← #
if prefix > 0 and suffix > 0 then
    stem_list[temp][0] ← prefix
    stem_list[temp][1] ← suffix
    stem_list[temp][2] ← root
    stem_list[temp][4] ← prefix_str
    stem_list[temp][5] ← suffix_str
return stem_list
end function
```

The morphological analyser was successful in finding the roots of 270,468 words and failed in analysing 292,216 words. Algorithm 3.4 details the steps undertaken to determine the validity of a given prefix-stemTemplate-suffix combination.

The combinations are produced by generating all possible prefix-stem-suffix templates as long as the stem part is at least 2-character long. The training knowledge was used to learn templates that were used to generate stems from roots. When a matching template is found, the root, and the stem are returned. It is worth mentioning that whenever Xerox Analyses generated more than one possible solution for any given word, all possible solutions were used for training

Sebawai, and their probabilities were assumed to be independent, which could distort the statistics, and force the system to produce roots that are not related to the word in question [19].

3.3 Light Stemmers

The inflectional part of the Arabic morphology is needed for this task to be accomplished correctly. A light stemmer handles the removal of prefixes, and suffixes to produce a stem. With certain stemmers, such as light10, it strips off a small set of, or the most frequent, affixes, and not the complete list. The stem has been proven in AIR experiments that this form is the optimal indexing term in terms of performance [6; 44].

3.3.1 BAMA

Tim Buckwalter was one of the three developers who were behind what is known now as the Xerox Arabic Morphological Analyser, along with Kenneth Beesley, and Stuart Newton. They implemented the Two-Level Morphology idea on the Arabic language [12]. Tim later left Xerox, and made public his own Arabic Analyser, which he called Buckwalter Arabic Morphological Analyser (BAMA) [17].

BAMA shares many features with the Xerox version, segmenting words into prefix-stem-suffix approach is one example, and also dividing the lexicon into prefixes, stems, and suffixes in order to establish the relationships between different morpheme categories [16]. That enabled Buckwalter to create the compatibility tables for the prefixes with the stems tableAB, the stems with the suffixes (tableBC), and the prefixes with the suffixes (tableAC).

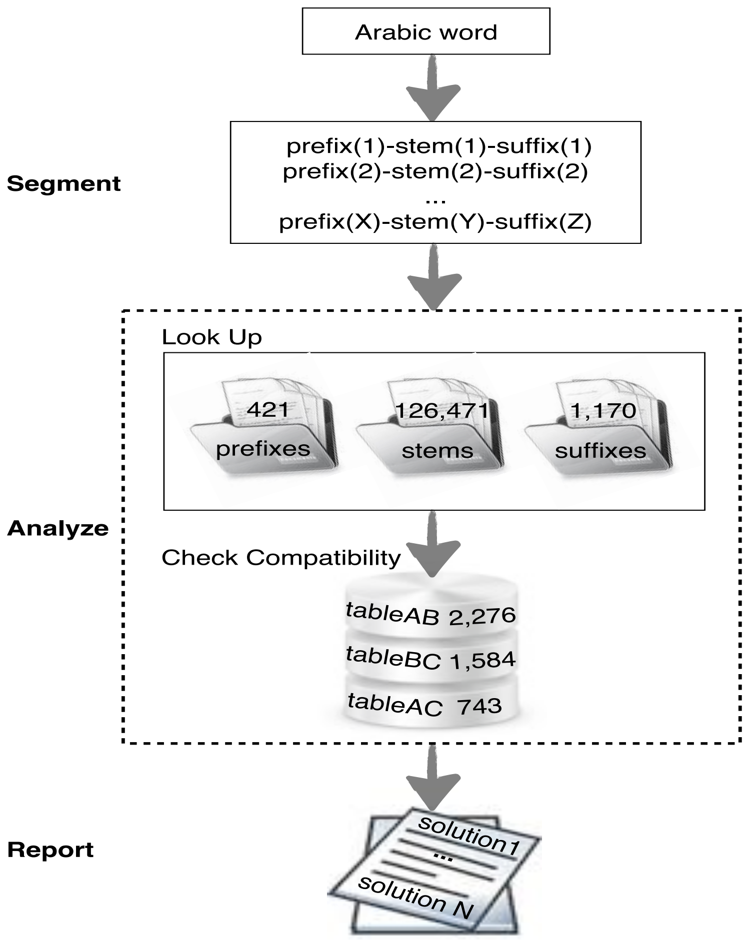


Figure 3.2: Buckwalter Arabic Morphological Analyser Main Steps.

BAMA is one of the most widely used analysers within the scientific community. It is the analyser, which was employed by the University of Pennsylvania Arabic TreeBank (PATB) project in order to annotate a large collection of news articles and produce segmented corpora, such as TREC-11 Arabic corpus, for research purposes [31; 47; 48].

BAMA consists of three lexicons: 421 prefixes, 126,471 stems, and 1,170 suffixes. There are also three compatibility tables; prefix-stem (tableAB), stem-suffix (tableBC), and prefix-suffix (tableAC). The tables AB, BC, and AC contain 2,276, 1,584, and 743 entries respectively. Once a word is read, the system goes through three major steps before reporting the possible solution(s) as Figure 3.2 depicts.

3.3.1.1 Segment Word

A word in the BAMA lexicon consists of a prefix of length 0 to 4 characters, a stem of length 1 to the maximum number of characters in the input word, and a suffix of length 0 to 6 characters.

Table 3.4: BAMA Segments for the Word "wAldyn"

Number	Segment			Number	Segment		
	Prefix	Stem	Suffix		Prefix	Stem	Suffix
1	∅	wAldyn	∅	11	w	A	ldyn
2	∅	wAldy	n	12	wA	ldyn	∅
3	∅	wAld	yn	13	wA	ldy	n
4	∅	wAl	dyn	14	wA	ld	yn
5	∅	wA	ldyn	15	wA	l	dyn
6	∅	w	Aldyn	16	wAl	dyn	∅
7	w	Aldyn	∅	17	wAl	dy	n
8	w	Aldy	n	18	wAl	d	yn
9	w	Ald	yn	19	wAld	yn	∅
10	w	Al	dyn	20	wAld	y	n

As depicted in Algorithm 3.5, the word is decomposed into all possible prefix-stem-suffix combinations subject to, the prefix length is ≤ 4, the stem length is ≥ 1, and the suffix length is ≤ 6.

BAMA starts with a prefix of length 0, then it attempts to find all possible stem-suffix combinations. The stem-suffix combinations for each prefix include, all possible stems whose length is ≥ 1, and a total of seven 7 suffixes if possible. So the maximum number of allowed segments here is 35, and that is equal to the 5 possible prefixes times the seven possible suffixes.

Algorithm 3.5: BAMA "segmentword" Function

Input: A word (str) of length ≥ 2.

Output: A list of all possible word segmentations.

```
function segmentword(str)
    segmented ← createlist()
    prefixLen ← 0
    while prefixLen ≤ 4 do
        prefix ← substr(str,0,prefixLen)
        stemLen ← strLen - prefixLen
        suffixLen ← 0
        while stemLen ≥ 1 and suffixLen ≤ 6 do
            stem ← substr(str,prefixLen,stemLen)
            suffix ← substr(str,prefixLen+stemLen,suffixLen)
            addsegment(segmented,prefix,stem,suffix)
            stemLen ← stemLen - 1
            suffixLen ← suffixLen + 1
        prefixLen ← prefixLen + 1
    return segmented
end function
```

Table 3.4 shows the possible prefix-stem-suffix combinations for the surface word "wAldyn" (والدين) meaning "*parents*". The BAMA segment function generates twenty possible prefix-stem-suffix combinations, one of which is our aim, that is segment number 3. It is to be noted that the "segment" algorithm pays no consideration to the validity of the segments because this step is undertaken by the "analyze" function.

3.3.1.2 Analyze

For each possible segment generated by the segmentation phase, the dictionary is checked for the validity of its prefix-stem-suffix combination. If they exist in the lexicon, then BAMA moves to the next phase of the analysis as shown in Algorithm 3.6 . Now the possible prefix-stem-suffix template is determined, the analyser goes through the items in the compatibility tables to check first for the legality of the prefix with the stem, and if that is allowed, then it checks the stem with the suffix, and the prefix with the suffix compatibility.

For instance, the definiteness marker "Al" "*the*", and the indefiniteness suffix"A" are not compatible, because the presence of one of them by definition excludes the other. When all combinations are compatible, an analysis report is produced listing the possible solutions. Adding the short vowels to the Arabic input letters leads to a single solution, while omitting them would lead to producing more possible solutions, and the user has to choose the correct analysis manually.

Algorithm 3.6: BAMA "analyze" Function

Input: A list of all possible word segments.

Output: A list of one or more solutions.

function analyze()

Algorithm 3.6: BAMA "analyze" Function – continued

```
        this_word ← nextToken(input)
        solutions ← createlist()
        count ← 0
        preLexicon ← Loadprefixes()
        stemLexicon ← Loadstems()
        sufLexicon ← Loadsuffixes()
        segmented ← segmentword(this_word)
        for segmentation ∈ segmented do
            prefix ← getPrefix(segmentation)
            stem ← getStem(segmentation)
            suffix ← getSuffix(segmentation)
            if prefix ∈ prefix_hash then
                if stem ∈ stem_hash
                    if suffix ∈ suffix_hash
                        for prefix_value ∈ preLexicon[prefix] do
                            catA ← getprefixcategory(prefix_value)
                            for stem_value ∈ stem_hash[stem] do
                                catB ← getstemcategory(stem_value)
                                if catA , catB ∈ tableAB
                                    for suffix_value ∈ suffix_hash[suffix]
                                        catC ← getsuffixcategory(suffix_value)
                                        if catA , catC ∈ tableAC
                                            if catB,catC ∈ tableBC
                                                count++
                                                add(solutions,count,prefix,stem, suffix)
        return solutions
end function
```

3.3.2 Light10

Based on the fact that "Considerable research on stemming and morphological analysis is amassing for the Arabic language, but no standard IR-oriented algorithm has yet emerged", a team lead by Leah Larkey at the University of Massachusetts,in research supported in part by the Centre for Intelligent Information Retrieval, has conducted an experiment on AIR, from which emerged several stemmers [44].

One of these is a very well known and called light10 stemmer, The number 10 here simply indicates the number of suffixes that light10 has in its lexicon. The University of Massachusetts team also ran a statistical experiment on TREC-10 data collection in order to categorise keywords based upon the stem classes. A clustering-algorithm based on removing vowels from Arabic words was implemented using co-occurrence analysis produced stem classes that were better than no stemming but inferior to good light stemming or morphological analysis.

The TREC-2001 Arabic corpus was chosen to run their experiments on, with 25 queries from TREC-2001 and 50 queries from TREC-2002 for a total of 75 queries. They stemmed all tokens before indexing as well as the queries keywords, and they ran light10 against Khoja root stemmer.

The results showed that the light10 performed better than Khoja root-based stemmer, and they concluded that light stemming produces better indexing terms than root-based stemmers. They reported that the light10 stemmer was more effective for Cross-Language Information Retrieval (CLIR) than a morphological stemmer, which tried to find the root.

Algorithm 3.7: Light10 "stem" Function

Input: An Arabic word of length > 3.

Output: Length of word after prefixes and suffixes removal.

```
function stem(word, length)
    length ← stemPrefix(word,length)
    length ← stemSuffix(word,length)
    return length
end function
```

3.3.2.1 stemPrefix

The light10 stemmer has the following seven prefixes with this order; "Al", "wAl", "bAl", "kAl", "fAl", "ll", and "w". The first six prefixes are nominal, the definiteness marker "*Al*" and its derivatives. The last prefix is the conjunction particle "w" "*and*", which can be used for both nouns and verbs. As depicted in Algorithm 3.8, the procedure goes through the list of prefixes in a sequential manner, if a prefix matches the start of the word, then it gets stripped off if it passes the validation conditions.

Algorithm 3.8: Light10 "stemPrefix" Function

Input: An Arabic word of length > 3.

Output: The length of the word after prefixes are removed.

```
function stemPrefix(word, length)
    for prefix ∈ prefixes do
        if startsWithCheckLength(word,length,prefix) then
            return deleteN(word, 0, length, prefix_length)
    return length
end function
```

The light10 stemmer attempts to impose two constraints on the prefix removal procedure as illustrated in Algorithm 3.9. The first condition is intended for the conjunction particle "w" "*and*" because it is the only prefix whose length is equal to 1. It makes sure that, after removing the conjunction particle "w" , the remaining number of characters is at least 3. The second guarantees that after stripping any of the first six prefixes off, there exists a stem of length ≥ 2.

Algorithm 3.9: Light10 "startsWithCheckLength" Function

Input: An Arabic word of length > 3.

Algorithm 3.9: Light10 “startsWithCheckLength” Function – continued

```
Output:  True if the word starts with the prefix, false otherwise.
function startsWithCheckLength(word, length, prefix)
    if prefix_length = 1 and length 4 then
        return false
    else if length < prefix_length + 2 then
        return false
    else
        for i = 0 to prefix_length do
            if word[i] ≠ prefix[i] then
                return false
        return true
end function
```

The light10 prefixes list does not contain any verbal prefixes, “y” or “t” for instance, which hinders its ability to stem verbs, and as a consequence jeopardizes the stemmer’s performance. Furthermore, If the prefix “w” had been placed in the first cell of the list, there would have been no need for the second prefix “wAl”, because removing the first two, “w” and “Al” would be equal to removing the prefix “wAl” itself.

3.3.2.2 stemSuffix

The same procedure is followed when removing suffixes as with prefixes. As Algorithm 3.10 shows, the light10 runs through the list in a sequential order and removes any matching suffix attached to the word. The Light10 suffixes are stored in an array with the order; “hA”, “An”, “At”, “wn”, “yn”, “yh”, “yp”, “h”, “p”, and “y”.

Algorithm 3.10: Light10 "stemSuffix" Function

Input: An Arabic word of length > 3.

Output: Length of word after suffixes are removed.

```
function stemSuffix(word, length)
    for suffix ∈ suffixes do
        if endsWithCheckLength(word,length,prefix) then
            length ← deleteN(word, length − suffix_length, length, suffix_length)
    return length
end function
```

The order of suffixes list would have a negative impact on the stemming process, as Table 5.25 shows, because suffixes are removed sequentially while the list order does not reflect the appropriate sequence of Arabic suffixes. Strictly speaking, the list should be ordered as; "h", "y", "hA", "yh", "yp", "An", "At", "wn", "yn", "p", so that the stemmer can resolve a combination of suffixes such as "Ath". But with the current logical flow of the stemmer, it would fail in such cases because it would check for the suffix "At" before it reaches the suffix "h", and that is in conflict with the Arabic rules pertaining to how suffixes are grouped together to form new suffix combinations.

Algorithm 3.11: Light10 "endsWithCheckLength" Function

Input: An Arabic word of length > 3.

Output: True if the word ends with the suffix, false otherwise.

```
function endsWithCheckLength(word, length, suffix)
    if length < suffix_length + 2 then
        return false
    else
        for i = 0 to suffix_length do
            if word[length − suffix_length + i] ≠ suffix[i] then
                return false
        return true
end function
```

The only validation step undertaken by the light10 stemmer when removing suffixes is merely making certain that the remaining number of characters is at least 2.

3.3.3 MADAMIRA

MADAMIRA is the combination of two tools in Arabic Natural Language Processing (NLP): MADA and AMIRA [61]. MADA is a system for Morphological Analysis and Disambiguation for Arabic, which was built upon BAMA with the addition of a selection process. The main task of MADA is, given raw Arabic text, to produce as much linguistic information as possible about each word in the text, thereby reducing or eliminating any ambiguity surrounding the word [33]. It then examines the generated analyses, and selects the best that fits the current context of the word through a process that uses support vector machines modeling 19 morphological features as classifiers [32].

AMIRA is a system that was developed by Mona Diab by adapting tools originally built for the English language and applying them on Arabic [24]. It performs three main tasks; tokenisation, part-of-speech tagging, and Base Phrase Chunking (BPC). AMIRA uses support vector machines with supervised classifiers, and it requires no prior knowledge of Arabic Morphology [25]. The tool had been developed by casting each of the morphological features as a classification problem. They system was trained on 750k Arabic words gathered from the Arabic Tree Bank (ATB) [48].

MADAMIRA is limited to both BAMA's ability and the training data in producing the stem. For instance, the word "fAx$whm" (فاخشوهم) was supposed to produce the stem "Ax$" (اخش), neither MADA nor AMIRA was able to produce the stem for this word. This word is somewhat tricky because it contains the prefix "f" (ف), and two suffixes; the subject pronoun "w" (و) and the object pronoun "hm" (هم). Furthermore, the vowel "y" (ي), which is part of the root “x$y” (خشي) was assimilated due to the presence of the suffix “w” (و). As mentioned in Table 3.2, multi-prefix and multi-suffix words pose a problem for BAMA, and this in effect had led to “NO-ANALYSIS” produced by MADA. As for AMIRA, it is more likely that this word was not included in the training data, which hindered AMIRA’s ability to generate the correct analysis.

3.4 Conclusion

We presented the Arabic stemmers that have been used in recent AIR research including TREC experiments. As it was shown, every stemmer has its own limitations, starting with the heavy cost of building the manually-constructed lexicon and maintaining it, the over-stemming issue with the light stemming approach if done without considering the Arabic morphology, and finally the root stemming approach where the affixes are removed from the word with no distinction given to the pattern, or to the multiple-affix combinations. It is our aim to overcome such limitations by imposing specific morphological rules on the stemming procedure that would lead to better results and enhance AIR performance.

Chapter 4

Enhancing Arabic Stemming via Morphological Analysis

A good lexicon is the cornerstone for any NLP application. With highly inflective languages, such as Arabic, it is appropriate to employ an efficient lexicon in text searching to save effort, and space. According to Hull's assessment of IR stemming algorithms "most stemmers operate without a lexicon and thus ignore word meaning, which leads to a number of stemming errors" [36]. One of the desired characteristics of an efficient lexicon is that it gives coverage for the language vocabulary using a small number of resources. In this chapter, we show how a concise morphological lexicon can be inferred from the Quran, and in turn be employed in the root stemming process.

We first present previous Quranic studies, and try to refine their methodologies to suit the Modern Standard Arabic (MSA) form, which is similar to Classical Arabic, but with one distinction, which is that MSA has no short vowels. The next two sections touch upon previous Quranic studies and their utilisation for educational purposes. In Section 4.1 the lexicon of the University of Haifa is presented, and we show how they leveraged the Xerox Finite-State Toolbox (XFST) to tag the Quran words in order to learn different morphological features that can be used for learning purposes.[1]

In Section 4.2, we explain how the University of Leeds Quranic corpus was de-

[1] http://cl.haifa.ac.il/projects/quran/index.shtml.

veloped, and also the methodology that was followed in constructing the lexicon is detailed.[2] In section 4.3, we present our customised methodology for constructing the lexicon, and show how that leads to the development of the Simple Arabic Stemmer (SAS), which outperforms state-of-the-art root stemmers, as will be shown in Chapter 5 [5].

In addition, we analyse in Section 4.4.2.3 some of the most commonly used patterns in the Arabic language, namely the participles and the broken plurals, and explore the relationship between the BP patterns and their singular forms (i.e., "Swt" (صوت), ">SwAt" (أصوات) meaning "*voice*", "*voices*").

4.1 Haifa

The linguistic group within the Computer Science at University of Haifa in a collaboration work with the Department of Arabic Language and Literature developed a computational system that is capable of querying the Quranic text by words as well as other linguistic features, such as patterns, roots, etc.

This idea was initiated to serve the scientific community by establishing a system to query and analyse the stylistic phenomena of the holy text, in addition to enabling them to take an in-depth look into how the Quranic phrases are structured linguistically [26]. They utilised Xerox Finite-state Toolbox (XFST) to annotate the Quranic words using a semi-automated methodology that goes through several stages, some of which involve manual creation of certain lists, and automatic extraction of other input data [13]. The objective was to build a lexicon that contains all of the Quranic words with their linguistic features, so that they could be searched, queried, and presented for learning purposes. The lexicon was divided into three types of words: closed-class (stop words), nominal bases, and verbal bases. Figure 4.1 demonstrates the multi-stage approach they

[2] http://corpus.quran.com.

employed in constructing their lexicon.

They manually compiled a list of a few hundred closed-class words (stop words) extracted from Abd al-Baaqii Indexed Quranic Words [57]. The same procedure was applied to nominal bases, where they gathered around 2500 from the same lexicon.

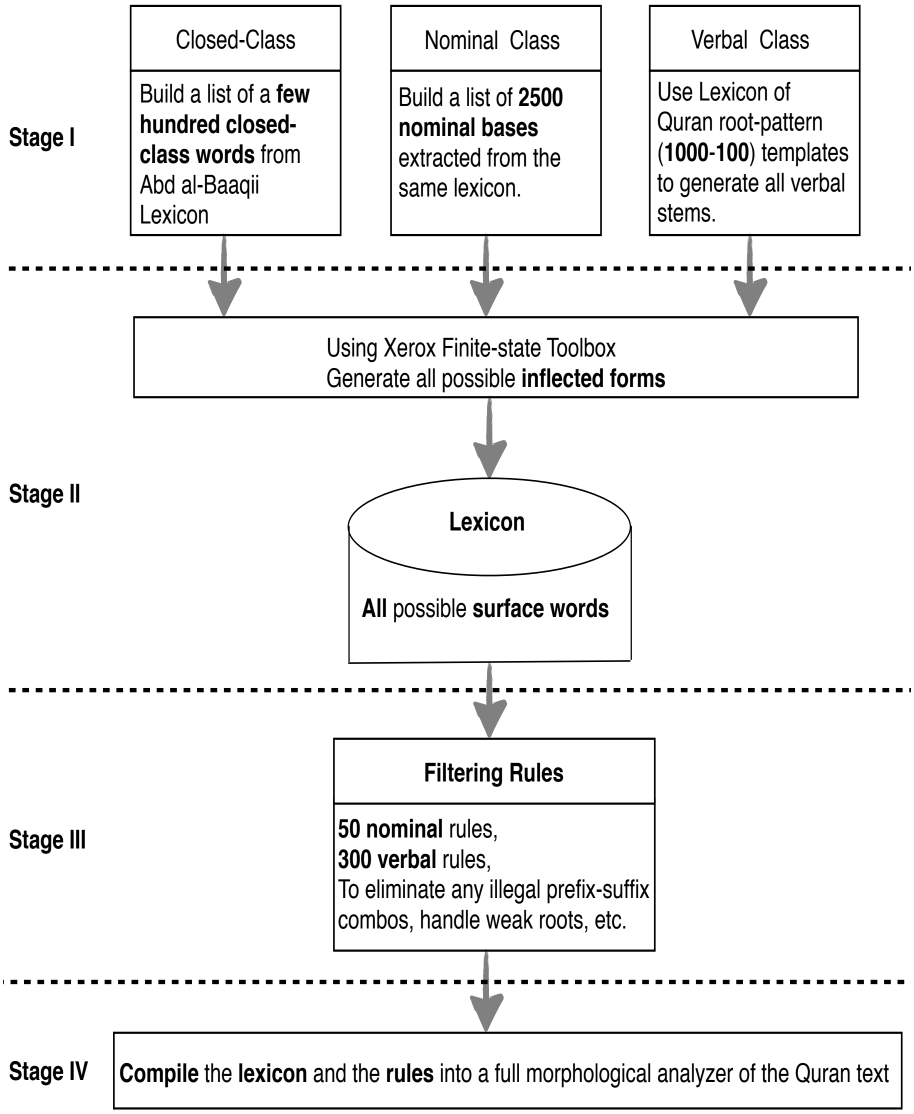


Figure 4.1: Haifa Lexicon Construction Methodology.

In the case of verbal bases, the process was not as straightforward as the other two categories, as they automatically extracted 1000 roots, and 100 verbal patterns from a Quranic lexicon to produce 100000 surface words, most of which are not in the corpus and had to be eliminated during the filtering process. They also manually encoded irregular forms, such as broken plurals, and proper names into the lexicon.

They utilised XFST to generate all possible inflected forms of what they prepared in the previous stage. It is worth mentioning that they used the toolbox to avoid annotating the Quran manually. This involves gathering all prefixes and suffixes of the Arabic language, and running XFST to produce all possible prefix-stem-suffix combinations.

The team dealt with the peculiarities of some nominal categories (e.g. broken plurals) with brute-force encoding of the irregular forms in the lexicon "since we are mostly concerned with a closed corpus here, this is a reasonable solution" [26]. Accordingly and as a result of producing all possible surface forms, there was an over-generation problem, for instance, the verbal bases would produce 100000 surface forms. This number of words was greater than the number of all of the Quranic words combined, and yet that was before the generating of the inflected forms.

The filtering rules were stipulated to eliminate any redundancies, any illegal prefix-suffix groupings, and also to resolve issues pertaining to weak roots. They created 50 nominal-bases rules and 300 verbal-bases rules to account for excessive generated surface words from the previous stage. For example, the lexicon would generate all combinations of the prefix "k" ("*as*") with all the nominal plural suffixes, whereas only the suffix "yn" (plural.masculine.genitive) is allowed. Therefore, a rule has to be written to eliminate other ill-formed combinations from the lexicon.

The last stage involves compiling the lexicon and the filtering rules to produce the full morphological tagging of the Quran words [76]. The lexicon consisted of several morphological attributes including the root, pattern, number, gender, and case. For example, a record of the word "AlrHym" QI (1.1.4) in the Haifa lexicon contained some information as shown in Figure 4.2.

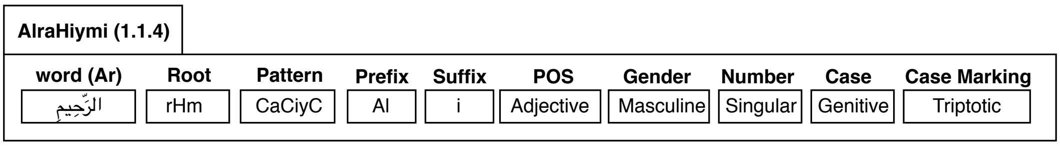


Figure 4.2: Haifa Word's Morphological Record.

Since it was nearly impossible to evaluate all the produced surface words, Haifa only tested their results partially. To measure the accuracy of their experiment, they manually annotated 1,248 words from the eighth chapter of the Quran. The system produced 1,440 possible solutions, 69 of which were incorrect, 205 were contextually wrong, and 1,162 were correct. Therefore, they recorded 93% recall, 80% precision and an f-measure of 0.86.

4.2 Leeds

Kais Dukes led a team as part of his PhD research to annotate and morphologically segment the Quran words. The aim was to provide researchers with a corpus that includes a large number of linguistic features, a thing that led to taking the Arabic grammar into consideration while building the lexicon. Their work has produced one of the online most comprehensive Quran lexicons [27]. Figure 4.3 depicts the multi-stage approach that was followed to morphologically annotate the Quran. The development of the lexicon went through two major phases, namely automatic construction and manual verification.

There had been three previously published Arabic corpora that were developed using BAMA as the core analyser with corpus-specific extensions: namely the Pennsylvania Arabic TreeBank (PATB), the Prague Arabic Dependency Treebank (PADT), and the Columbia Arabic Treebank (CATiB) [34; 47; 73]. Accordingly, the team employed the BAMA analyser to annotate the Quran words. The team had to extend the analyser to meet the specifics of the Quranic Classical Arabic language, and so three extensions were added; spelling, ranking and filtering. BAMA was originally built for the MSA, which differs slightly in spelling from the Classical Arabic language. The team had to make several modifications pertaining mainly to the orthographic variation of the "HAMZA" (glottal stop) consonant, and the multiple shapes of the "A" vowel.

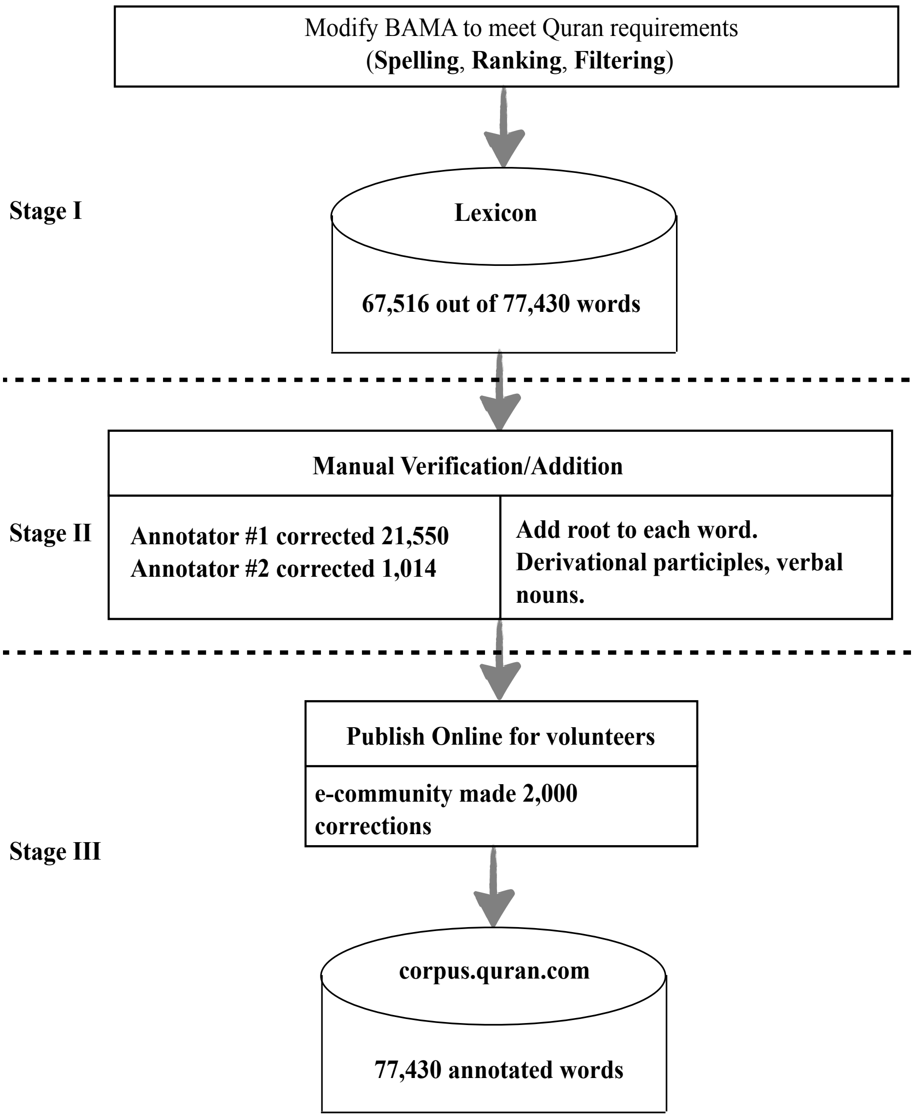


Figure 4.3: Leeds Quran Corpus Construction Methodology.

The filtering process was undertaken manually to eliminate any incorrect analysis. The ranking step was also carried out manually to order the possible analyses according to how close they are to the original text, with the closer matches ranked higher. The BAMA analysis with the highest rank is then selected as the correct part-of-speech for that word.

The Leeds team applied the automatic algorithm on the Quranic text, and the system produced 67,516 out of 77,430 words. Due to out-of-vocabulary errors in BAMA lexicon, the automatic results were then reviewed manually. Two annotators were assigned the task of verifying each word manually, with the second annotator reviewing the text after the initial set of corrections were made by the first. This resulted in the addition of 9,914 new words that were missing from the automatic generation, and the first annotator made 11,636 corrections to the existing analysis, while the second annotator made changes to 1,014 words.

There were certain features that had to be added manually; those include the missing verb voice (active/passive), the energetic mood for verbs, the interrogative "ALEF" prefix, identifying participles, verb forms, and disambiguating the "LAM" prefix. The team also added the root of each word, and this was processed automatically by extracting the root of each word from an online certified root list.[3] They measured the performance of their automatic algorithm, and recorded 72% recall, 83% precision and 0.77 f-measure.

After manually verifying the corpus, the team published it online for community volunteers to make corrections or suggest modifications. And that resulted in 2,000 corrections and certain level of confidence regarding the accuracy of the produced lexicon since so many users (average 1000/day) participated in the last step and gave their approval for the 77,430 annotated words. Figure 4.4 shows an example of a Leeds morphological record.

AlraHiymi (1.1.4)

word (Ar)	Root	Pattern	Prefix	Suffix	POS	Gender	Number	Case	Case Marking
الرَّحِيمِ	rHm	— — —	Al	i	Adjective	Masculine	Singular	Genitive	— — —

Figure 4.4: Leeds Word's Morphological Record.

[3] http://www.zekr.org.

As it can be observed, the major difference between the Haifa and Leeds lexical record is the absence of the morphological pattern from the Leeds record. The pattern represents a very valuable attribute due to its relation to the other words; for instance, the broken plurals of certain patterns would be useful in order to study the non-concatenative aspect of the morphology at an intensive level, bearing in mind that such studies cant be conducted without the existence of the pattern for each word in the lexicon.

4.3 Simple Stemmer Customised Methodology

The development of a Quran-based morphological lexicon is not a trivial task as was demonstrated by the two previous projects. The manual verification is an essential process, and ought to be undertaken in order for a researcher to ensure the accuracy of the lexical entries. In a similar approach to the two projects discussed above, a multi-stage development methodology was executed to reach a suitable lexicon. What distinguishes this approach is that it deals with words without the short vowels, and therefore the number of morphemes in the lexicon is shrunk to the limit. Figure 4.5 shows the different stages followed during the construction of the lexicon.

The SEQUITUR algorithm recognises repeated symbols in strings and compresses them to form what is called a rule [56]. It has been tested on holy texts previously, including a run on the Bible, and showed superior performance over known compressing algorithms. Table 4.1 shows an example of the sequence "abcdbcabcd".[4] When combining symbols to form rules, the algorithm operates with two constraints to watch for:

- no pair of adjacent symbols appears more than once (diagram uniqueness);
- every rule is used more than once (rule utility).

Note in number 6, the addition of the symbol "c" to the sequence triggers the action needed to maintain the first constraint, since the pair "bc" has appeared

[4]This is an exact copy of Table 1 on page 70 of the SEQUITUR paper.

previously. Therefore a new rule is created, which assigns the terminal symbols "bc" to the non-terminal symbol "A".

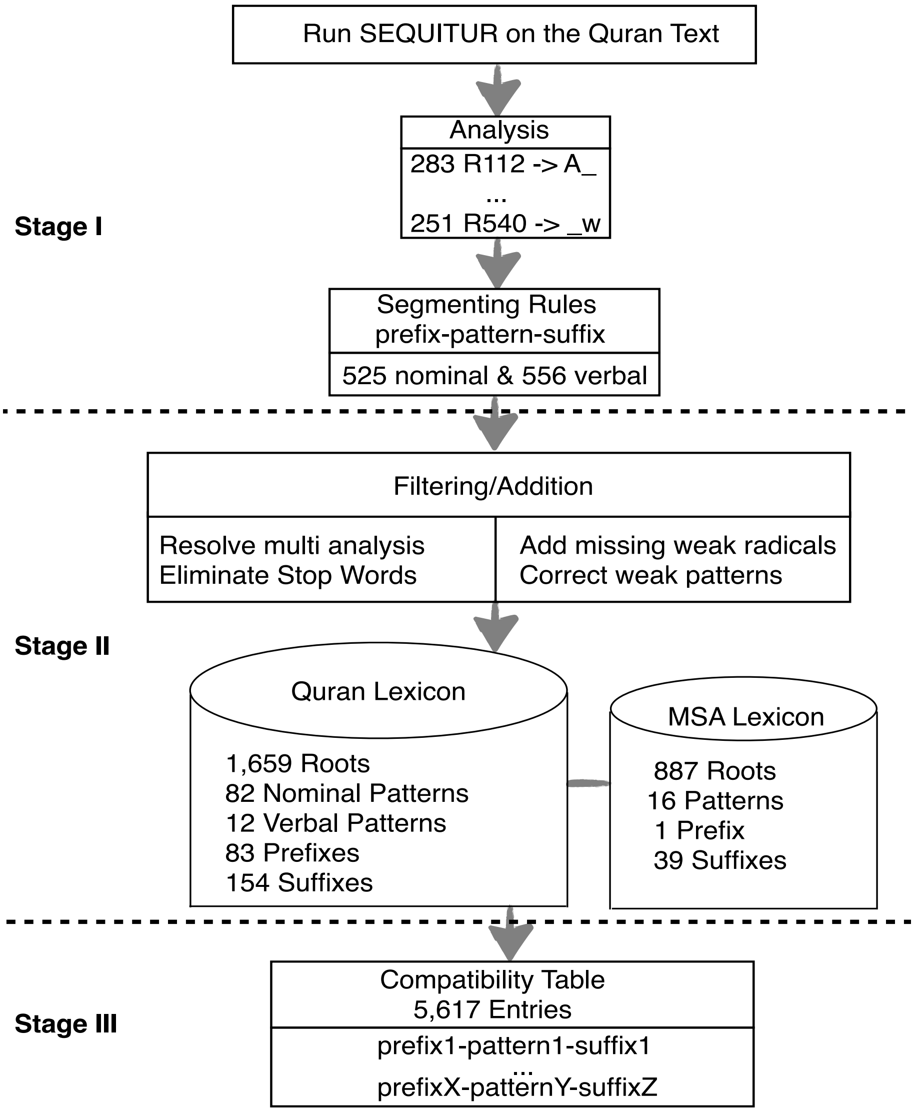


Figure 4.5: Customised Lexicon Construction Methodology.

The second constraint is depicted in number 10, where the enforcement of the first constraint causes the violation of the rule utility constraint for rule "B", because it is used only once.

Table 4.1: Operation of the Two Grammar Constraints, Digram Uniqueness and Rule Utility

number	string so far	resulting grammar	remarks
1	a	S $\rightarrow$ a	
2	ab	S $\rightarrow$ ab	
3	abc	S $\rightarrow$ abc	
4	abcd	S $\rightarrow$ abcd	
5	abcdb	S $\rightarrow$ abcdb	
6	abcdbc	S $\rightarrow$ abcdbc	bc appears twice
		S $\rightarrow$ aAdA	enforce digram uniqueness
		A $\rightarrow$ bc	
7	abcdbca	S $\rightarrow$ aAdAa	
		A $\rightarrow$ bc	
8	abcdbcab	S $\rightarrow$ aAdAab	
		A $\rightarrow$ bc	
9	abcdbcabc	S $\rightarrow$ aAdAabc	bc appears twice
		A $\rightarrow$ bc	
		S $\rightarrow$ aAdAaA	enforce digram uniqueness
		A $\rightarrow$ bc	aA appears twice
		S $\rightarrow$ BdAB	enforce digram uniqueness
		A $\rightarrow$ bc	
		B $\rightarrow$ aA	
10	abcdbcabcd	S $\rightarrow$ BdABd	Bd appears twice
		A $\rightarrow$ bc	
		B $\rightarrow$ aA	
		S $\rightarrow$ CAC	enforce digram uniqueness
		A $\rightarrow$ bc	
		B $\rightarrow$ aA	B is only used once
		C $\rightarrow$ Bd	
		S $\rightarrow$ CAC	enforce rule utility
		A $\rightarrow$ bc	
		C $\rightarrow$ aAd	

That results in deleting rule "B" from the grammar after moving its constituents "aA" to rule "C". The SEQUITUR algorithm was chosen to assist in locating the most repeated affixes (prefixes and suffixes) in the Quran, and then use these affixes in the segmenting templates to mark the word's morphological units. A word in Arabic consists of a prefix, a stem, and a suffix, so the output of the SEQUITUR algorithm is employed to specify where an affix starts and where it ends, leaving the stem part to be matched against a list of patterns to extract the root.

An XML version of the Quran text was downloaded from the University of Leeds Quran corpus. The Quran text is composed of 607,359 characters divided into 330,709 alphabetical letters (A,b, t, etc.), and 276,650 diacritics such as short vowels. Since the Quranic text is different from all other Arabic texts, in the sense that every letter is accompanied by a short vowel signifying its sound, a preprocessing step was undertaken to remove all diacritics before running the SEQUITUR algorithm.

The algorithm produced 17,619 rules after compressing the text by grouping any two adjacent symbols into a rule if they appeared more than once. Table 4.2 shows a sample output from running the SEQUITUR algorithm on the Quran text.

Table 4.2: A Sample Output after Running SEQUITUR on the Quran text

Number	Frequency	Rule	Constituents
1	283	R112	A _
2	57	R2294	_ A
3	164	R139	R2294 L
4	111	R403	wn
5	264	R1573	R403 _

The first row of the sample shows that an "A" in the word's final position is the most frequent suffix.[5] That is no surprise as it is one of a few suffixes which

[5]The appearance of the "_" symbol after a consonant marks the end of the word.

can be used with both nouns: as an indefiniteness marker for the accusative case, and verbs as a subject pronoun (masculine.dual.3rd person).

In the second row, R2294 represents the start of the word marker "_" and the letter "A", or any word that starts with the vowel "A", whereas the third row is just the definiteness determiner prefix because it consists of R2294 and the letter "l" or "_Al". The fourth row contains R403, which is the two letters "wn", and the fifth row is R1573 representing R403 and the end of word marker "_", or the suffix "wn_", which is used with both verbs and nouns.

From that analysis phase, segmenting templates of the form prefix-pattern-suffix were created in order to match the Quran words against. Since the stem is merely a pattern template such as "muCCiC" and a root's radicals filling the Cs slots, extracting the root's radicals would be a matter of matching letters that correspond to the Cs positions.

The most frequent affixes derived from the previous stage were attached to various nominal and verbal patterns so that input words can be segmented according to those boundaries. For example to recognise words that have, the definiteness determiner "Al" ("*the*"), the nominal pattern "muCCiC", and the suffix "wn". After removing the short vowels, the following template can be utilised for that purpose:

Al+m C C C+wn
R139+m C C C +R1573

One such example of a word matching this template after removing the short vowels is "Al+mslm+wn" ("*the Muslims*") with the root "slm". Each letter is given an integer number indicating its index within the alphabets list, "HAMZA" being the first has 0 as its integer value, and "SUKUN" being the last is assigned 43.[6] The root radicals were represented in the templates as negative integers while affixes (prefix, suffix, infixes) were given their corresponding positive integer values (e.g., 30 for the letter "m", 33 for the vowel "w", etc.). An example of a template representing the word "AlrHym" is shown in Table 4.3.

The reason for this process is two-folds, firstly, the root and the pattern of each word have to be present in the lexicon, and secondly, the number of patterns

[6] We used the UTF numbering, see http://www.unicode.org/charts/PDF/U0600.pdf.

used in the Quran is quantified, and so that we can guess whether they are morphologically related. During this stage all patterns Quran were encoded with their affixes obtained from the SEQUITUR output.

Table 4.3: Segmenting Template Example

prefix		Radical 1	Radical 2	Infix	Radical 3
A	l	r	H	y	m
6	29	−1	−2	35	−3

All triliteral strong roots that are produced using the pattern CCyC and the definiteness determiner "Al" would be recognised by this template. To cover the Quran surface forms after removing the STOP words, we created 525 nominal and 556 verbal templates, "segmenting rules" as they are interchangeably referred to throughout the thesis.

The filtering/addition was the most time consuming stage, because every analysis had to be checked and verified. As there are words that could be represented with multiple analyses, each word was then manually checked to make certain it represented solely one valid morphological combination, and any extra analyses would be eliminated hence. For example, the word "farAga" (37.91.1) would match a verb and a noun. The verbal form consists of the conjunction particle "fa", when meaning "*then*", and the perfective verb (Form I) of the weak root "rwg", and so the result will be "farAga" (فراغ) meaning "*then he went unnoticed*". The nominal case would be composed of the pattern CaCAC, the root "frg", and the suffix "a" marking an accusative case (i.e., an object of a verb) to produce the word "farAga" meaning "*emptiness*". The former is the correct analysis after verifying it from the Quran text, and the correct answer is manually selected.

The morphological aspect of the holy text is our focus since we are mainly concerned with the impact of morphology on AIR performance. Therefore it was essential to add the root and the pattern for each word in the lexicon using the same numbering scheme mentioned above. All of the weak roots had to be checked and corrected manually for the vowels that tend to take other shapes, and eventually, the correct roots were encoded in the lexicon to ensure the accuracy and the integrity of the next phase.

We used a Quranic lexicon that was published in 2002 authored by a 25-member team led by Professor Ahmed Mukhtar Omar [60]. Since the Quran was studied from a purely linguistic perspective, the book is valuable in terms of its morphological content. This lexicon lists the Quran words according to their roots, and using the root, one can find all patterns used in the Quran for that particular root. Figure 4.6 shows the final lexical record for the same word "Alrahiymi" (الرّحيم) seen previously in Table 4.3.

AlraHiymi (1.1.4)

word (Ar)	Root	Pattern	Prefix	Suffix	POS	Gender	Number	Case	Case Marking
الرَّحِيم	rHm	CCyC	Al	$\emptyset$	Noun	−−−	−−−	−−−	−−−

Figure 4.6: SAS Word's Morphological Record.

The Quran lexicon contained 1659 roots, 94 patterns, 83 prefixes and 154 suffixes. Since the Modern Standard Arabic might include words that have no mention in the Quran, we extended our lexicon by stemming 6000 more words from the Saudi newspaper Alriyadh[7] and the Lebanese newspaper Alnahar web

[7] http://www.alriyadh.com.

archives.[8]

The new words were selected from the main articles in both newspapers, the existence of the morphological units that made up the word was our selection criterion. A new word would be a candidate for an entry into our lexicon if it contained a morphological unit such as a root, a pattern, a prefix, or a suffix that was not part of our lexicon. For instance the suffix "Atuhum" (اتهم) was not mentioned in the Quran, but it was added during this stage because it was found in a word within these articles. From this corpus, we were able to add to our initial lexicon 887 roots, 16 patterns, 1 prefix, and 39 suffixes.

4.4 Simple Arabic Morphological Lexicon

The lexicon is a major component of my solution as it is required in order to have an accurate morphological analysis of words.

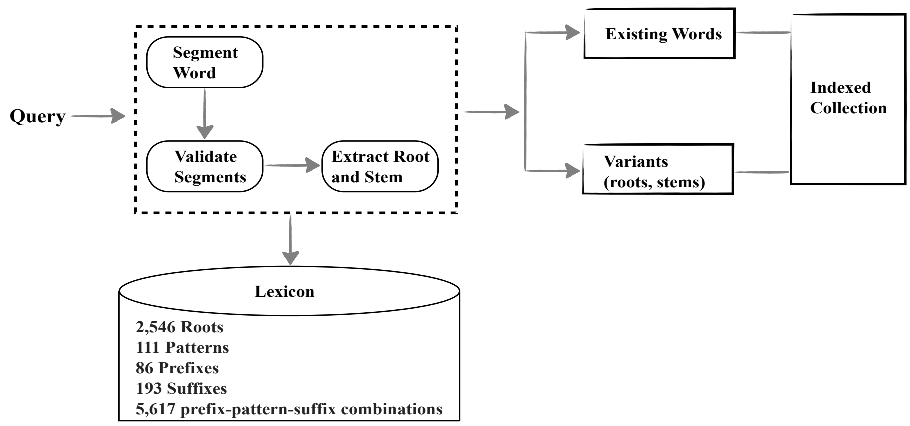


Figure 4.7: Proposed Solution Components.

[8] http://www.annahar.com.

As Figure 4.7 shows, there are five major components of the lexicon; roots, patterns, prefixes, suffixes, and legal combinations restricting prefix-pattern-suffix groupings to those present in the lexicon.

4.4.1 Roots

The root in the Arabic language represents the cornerstone of the lexicon, on which so many derived words can be based. The roots in the Quran are the units from which the true meaning of words can be recognised. Our analysis classifies roots into five types. The first is the Strong (S) type, and this by definition is any root whose radicals do not include any vowels. The second type is the Geminate (G), which includes any root whose second and third radicals are identical, or duplicated. The third type is the Single Weak, and it includes roots with a single vowel as one of their radicals, whether be the initial, the middle or the last radical. The fourth type is the same as the third except two of the radicals instead of one are vowels and it is called the Double Weak. The last and the fifth type does not occur in the lexicon very often, however it is one of the types that dictate certain phonological transformations when produced using nominal patterns, and it will be called All Weak.

Table 4.4: Quran Roots and their Category Distribution

POS	S	G	R1	R2	R3	R1R2	R1R3	R2R3	R1R2R3
Verb	495	86	64	128	141	4	18	20	1
Noun	751	115	106	189	148	11	23	20	2

In Table 4.4, the root's radicals are represented by R1, R2, and R3. The table breaks the double weak roots into initial, middle, and last to indicate which radical is the vowel. As the table shows, the number of roots used by nouns far exceeds that used with verbs. One interesting observation about all weak roots is that there is one weak root that was used very often with nouns, but never used with verbs, namely the root ">yy", from which the word "|yap" (آية) meaning ("*verse, mark*") is derived. There existed eleven quadriliteral roots that were used with verbs, and forty six roots with nouns.

4.4.2 Nominal System

Nouns in the Quran will be studied to learn how they are formed in terms of the root and pattern types. The Strong roots will not be analysed much deeper than their surface value because they offer no special rules as far as the production of the surface forms is concerned. This is not the case, however, with the other four types since they reflect the rules that will be learnt for the morphological validation phase.

Historically, there are two schools of Arabic grammarians; one school teaches nouns as the base word forms and verbs are derived from them. On the other end of the spectrum, there is another school whose scholars firmly believe that verbs are the base and nouns are merely derived from them. In all fairness, each school has its own reasons, and both schools are embraced by morphologists. The former approach is adapted in this thesis because of the following:

- In the lexicon, there are 82 nominal, and 12 verbal patterns.
- The number of roots in the nominal system exceeds that of the verbal by hundreds, which means that there are hundreds more roots that could be generated as nouns.
- Since this study is pertaining to Information Retrieval, it is a well-known fact that nouns are more informative than verbs when selected as search keywords.
- There are more phonological cases in the nominal system than in the verbal system, as vowels are altered and sometimes assimilated within a pattern that contains another vowel. This is observed more frequently with nouns since the number of patterns is greater.

Based on the above factors, the evidence is that the nominal system would tend to reveal more morphological information about the word, because it addresses both derivational and inflectional morphology on a much wider scale as will be shown in the subsequent sections.

4.4.2.1 Prefixes

Prefixes are very important in word structure because they bear grammatical features besides the change that the word has to undergo to reflect their presence. Nominal prefixes in our lexicon are divided into four groups; interrogation, conjunction, preposition particles and the definiteness determiner. There are cases in the Quran where the prefix "l" is employed to stress the importance of the word as an affirmative particle in addition to its usual function as a preposition particle. Therefore, the genitive suffix "yn" does not necessarily accompany the prefix "l", as it is clear from Table 4.5, where the nominative suffix "wn" is left in the word, which suggests that the affirmative particle "l" has no impact on the nominal case. It is worth mentioning the fact that this kind of language could only be observed in the Quran corpus or related literature, because it just sounds too classical and is considered out-of-date for contemporary MSA speakers.

Table 4.5: The Prefix "l" and its Usage in the Quran

Constituents	Ar. Word	QI	Gloss
lil*Akiryn	للذاكرين	11.114.14	for those who remember
lmunqalibwn	لمنقلبون	43.14.4	must be returning

The longest nominal prefix in the lexicon consists of four prefixes combined together. It starts with the interrogative particle ">", the conjunction "f" followed by the preposition "b", and finally the definiteness determiner "Al", as Table 4.6 shows.

Table 4.6: Four Prefixes Grouped in One Word

Constituents	Ar. Word	QI	Gloss
>+fa+bi+Al+bATil	أفبالباطل	16.72.16	Is it with the falsehood?

Also there are some rare combinations of prefixes that could only appear in the Quran vocabulary such as the prefix "l" grouped with the preposition particle "b" and the definiteness determiner Al, as in the first word of Table 4.7, or without the definiteness marker as in the second word.

Table 4.7: Rare Prefixes Grouping

Constituents	Ar. Word	QI	Gloss
la+bi+Al+mirSAd	لبالمرصاد	89.14.3	everything is watched
la+bi+sabyl	لبسبيل	15.76.2	it is by the path of

Table 4.8 shows the most popular nominal prefixes found in our lexicon, which contains fifty five prefixes.

Table 4.8: Nominal Popular Prefixes

Prefix	Freq.	Word	Ar. Word	QI
∅	3793	|bA'	آباء	24.31.27
Al	1160	Al|fAq	الآفاق	41.53.4
w	552	w|bA}hm	وآبائهم	21.44.4
b	360	b|bA}nA	بآبائنا	44.36.2
w+Al	304	wAl|SAl	والآصال	7.205.12
l	241	l|bA}him	لآبائهم	18.5.7
b+Al	136	bAl>bSAr	بالأبصار	24.43.38
l+l	93	lil>brAr	للأبرار	3.198.21
f	53	f>SHAb	فأصحاب	56.8.1
k	30	k>mvAl	كأمثال	56.23.1

Nouns with no prefixes constituted more than half of the lexicon entries. The definiteness determiner "Al" "*the*" comes next, being used 1,160 times, followed by the conjunction particle "w" "*and*", and then the preposition particle "b". What comes after that is the combination of the conjunction particle "w" and the definiteness determiner "Al", which has 304 occurrences. The preposition "b" ("*by*") and the definiteness determiner "Al" has occurred 136 times, compared to 22 for the preposition "k" ("*as*") and the definiteness determiner.

4.4.2.2 Suffixes

There are certain rules that have to be followed when dealing with nominal suffixes. One rule dictates that no suffix can be added after a possessive pronoun.

Table 4.9: Nominal Popular Suffixes

Suffix	Freq.	Word	Ar. Word	QI
∅	2730	|bA'	آباء	24.31.27
A	1013	>bkArA	أبكارا	56.36.2
p	631	|typ	آتية	20.15.3
yn	430	|x*yn	آخذين	51.16.1
hum	308	|bA'hum	آباءهم	23.68.9
wn	275	|xrwn	آخرون	25.4.12
h	270	>zwAjh	أزواجه	33.53.61
kum	207	|bA'kum	آباءكم	2.200.7
At	161	AlSAfinAt	الصّافنات	38.31.5
hA	135	>bSArhA	أبصارها	79.9.1

Moreover, there are certain prefixes that dictate the dual and plural nominal cases (i.e., preposition → genitive). The lexicon contains fifty nine suffixes, and Table 4.9 shows fifteen of the most popular suffixes.

4.4.2.3 Patterns

There are two nominal types that are used more than any other; namely the Active Participle and the Broken Plurals. The active participles are very popular in Arabic because it conveys more information since it represents the verb and the subject of the sentence. Therefore, by using only the active participle of the verb as an example, one can tell about the actor and the action (verb) at the same time. They are also called verbal adjectives in Wright's Grammar Book [81] (p:131), and could be derived from all triliteral verbs, so the division of presentation will be based on the forms of the triliteral verb from which the verbal adjective is derived.

Table 4.10 shows what is called Active Participles, which are used very often because they represent the verb as well as the subject. The first pattern in Table 4.10 is the active participle of the first verbal pattern in Table 4.32.

Table 4.10: Active Participles for the 10 Triliteral Verbal Patterns

Num.	Pattern	Stem	Gloss	Ar. Word	QI
1	CACiC	kAtib	writer	كاتب	2.282.14
2	muCaC~iC	mubad~il	modifier	مبدّل	6.115.7
3	muCACiC	mulAqykum	confronted	ملاقيكم	62.8.8
4	muCCiC	muslimA	Muslim	مسلما	12.101.19
5	mutaCaCCiC	Almutakab~ir	highest ego	المتكبّر	59.23.15
6	mutaCACiC	muta$AbihA	same look	متشابها	39.23.6
7	munCaCiC	munqalibwn	returners	مُنْقَلِبُونَ	7.125.5
8	muCtaCiC	muqtadirA	able	مُقْتَدِرًا	18.45.23
9	muCCaC~	muSfar~A	yellowish	مصفرّا	30.51.5
10	mustaCCiC	mustakbirwn	arrogants	مُسْتَكْبِرُونَ	16.22.11

The noun "kAtib" "*writer*" conveys the identity of the subject in addition to the type of action performed "writing", and that is why Classical Arabic speakers would rather use this type of nouns because a single word can bear two implicit features. Consider Table 4.11 for the number and the type of roots that have been used with this pattern.

Table 4.11: Form I Active Participle Roots Distribution

Root Type	Occurrence	Surface Form	Roots
Strong	1,461	480	195
Geminate	57	24	14
Single Weak	551	246	106
Double Weak	43	26	11
Total	2,112	776	326

Table 4.12 shows active participles for the two quadriliteral verbal patterns. It is apparent that all active participles, but the 9th, including quadriliteral patterns end with the segment "CiC".

Table 4.12: Active Participles for the 2 Quadriliteral Verbal Patterns

Num.	Pattern	Stem	Gloss	Word	Ar. Pat.	QI
11	muCaCCiC	bimuzaHziHihi	mover	بمزحزحه	مفعلل	2.96.17
12	muCCaCiC˜	muTma<n˜	assured	مطمئنّ	مفعلّ	16.106.11

The nominal passive participles play the same role as the active participles except instead of revealing the subject, they reveal the object. The first verbal form has the passive participle as maCCwC, so for the root "ktb", "maktwb" "*written*" is the stem as in Table 4.13.

Table 4.13: Form I Passive Participle

Num.	Pattern	Stem	Gloss	Word	Ar. Pat.	QI
13	maCCwC	maktwbA	written	مكتوبا	مفعول	7.157.8

For the the rest of the verbal forms, the 9th is an exception again, share the same pattern template with the active participle with the last short vowel changed to an "a" instead of an "i" as in Table 4.14, the passive participle ends with the segment "CaC" instead of "CiC".

Table 4.14: Form II Passive Participle

Num.	Pattern	Stem	Gloss	Word	Pat.	QI
14	muCaCCaC	musal˜amap	handed to	مسلّمة	مفعّل	4.92.17

The next category is the infinitive, which according to Wright's definition "expresses the action, passion, or state indicated by the corresponding verb, without any reference to object, subject, or time" [81].

Table 4.15: Infinitives for the 10 Triliteral Verbal Forms

Num.	Pattern	Stem	Gloss	Ar. Word	QI
15	CiCACap	tijArap	commerce	تجارة	4.29.13
16	taCCiyC	tabdiylA	change	تبديلا	33.23.18
17	CiCAC	AlHisAb	accountability	الحساب	38.16.8
18	<CCAC	Al<slAm	Islam	الإسلام	3.19.5
19	taCaC~uC	taqal~ub	turning	تقلّب	2.144.3
20	taCACuC	watafAxur	bragging	وتفاخر	57.20.8
21	AnCiCAC	AnfiSAm	breaking	انفصاما	2.256.20
22	ACtiCAC	AxtilAf	difference	اختلافا	4.82.12
23	ACCiCAC	ASfirAr	yellow	اصفرار	
24	AstiCCAC	AstikbArA	arrogance	استكبارا	35.43.1

Table 4.15 shows the 10 infinitive nouns for the 10 triliteral verbal patterns in table 4.32 in the same order. So pattern number 15 in Table 4.15 is the infinitive form for pattern number 1 in Table 4.32 and so on.

Table 4.16: Infinitives for the Triliteral Verbal Forms

Num.	Pattern	Stem	Gloss	Ar. Word	QI
25	CuCCAn	gufrAnka	pardon	غفرانك	2.285.24
26	taCCiCap	ta*kirap	reminder	تذكرة	20.3.2
27	CaCaCwt	malakwt	absolute power	ملكوت	23.88.4
28	tiCCAC	tibyAnA	clarity	تبيانا	16.89.18
29	CACiCp	AlfaHi$p	obscenity	الفاحشة	4.15.3
30	CaCACyp	waEalAnyp	disclosure	وعلانية	113.22.12

There are verbs that have more than one infinitive, which explains why we have 6 more patterns that have been included in the lexicon. Table 4.16 shows the extra infinitives for the triliteral verbs. For instance, pattern number 25 in Table 4.16 can also be used besides pattern number 14 in Table 4.15 to form the infinitive for pattern number 1 in Table 4.32, depending on the root that is used in generating the verbal form.

Table 4.17 shows the two quadrilateral infinitives representing the two verbal patterns in Table 4.33.

Table 4.17: Infinitives for the 2 Quadrilateral Verbal Forms

Num.	Pattern	Stem	Gloss	Word	Pattern	QI
31	CaCCaCap	zalzalap	shaking	زلزلة	فعللة	22.1.7
32	AiCCiCCAC	AiTm< inAn	relief	اطمئنان	افعلّال	

The nominal system consists of many adjectives, Table 4.18 shows three patterns that are called verbal adjectives because they usually indicate the state of the subject, and a verb of some sort is used in their generation. The first pattern in the table is used everyday more frequently than any other, because when it is combined with the root "mwt", they produce the word "mayyit" ("dead"), note how the vowel "w" is transformed into a "y" due to the vowel "y" that is part of the pattern.

Table 4.18: Verbal Adjectives

Num.	Pattern	Stem	Gloss	Word	Pattern	QI
33	CaCyC	karym	generous	كريم	فعيل	8.4.11
34	CaCwC	gafwr	forgiving	غفور	فعول	2.173.24
35	CayCaC	may˜at	dead	ميّت	فيعل	39.30.2

The patterns in Table 4.19 produce the intensive adjectives, which indicate, according to Wright "a very high degree of the quality which their subject possess".

Table 4.19: Intensive Adjectives

Num.	Pattern	Stem	Gloss	Word	Pattern	QI
36	CuC~wC	Alqud~ws	holy	القدّوس	فعّول	59.23.9
37	CaC~AC	Sab~Ar	patient	صبّار	فعّال	42.33.14
38	CyCwC	Alqy~wm	responsible	القيّوم	فيعول	2.255.7
39	miCCyC	miskynA	poor	مسكينا	مفعيل	90.16.2

The nouns denoting instruments are frequently used in Arabic, which usually refer to tools and instruments, one recent added noun of instrument to the Arabic lexicon is "HAswb" (".haAsuwb") ("*computer*"). The word "HAswb" is derived through the intersection of the pattern "CACwC" and the root "Hsb" ("*having to do with calculating*"). Two major patterns dominate this category shown in Table 4.20

Table 4.20: Triliteral Nouns of Instrument

Num.	Pattern	Stem	Gloss	Word	Pattern	QI
40	CACwC	AlnAqwr	trumpet	النّاقور	فاعول	74.8.4
41	miCCAC	wAlmiyzAn	scale	والميزان	مفعال	11.84.18

The nouns of individuality are expressing the tangible things all around us, for instance "qamar" ("moon"). Although there exist many patterns within this category, Table 4.21 merely shows two sample patterns of such a category.

Table 4.21: Triliteral Individuality Nouns

Num.	Pattern	Stem	Gloss	Word	Pattern	QI
42	CawCaC	Alkawvar	river in heavens	الكوثر	فوعل	108.1.3
43	CaCaCap	baqarap	cow	بقرة	فعلة	2.67.10

Table 4.22 shows quadriliteral nouns of individuality.

Table 4.22: Quadriliteral Individuality Nouns

Num.	Pattern	Stem	Gloss	Word	Pattern	QI
44	CuCCwC	kAlEurjwn	branch	كالعرجون	فعلول	36.39.6
45	CaCCaCyC	zamharyr	cold	زمهريرا	فعلليل	76.13.10
46	CiCCAC	qirTAs	paper	قرطاس	فعلال	6.7.6
47	CiCCiyC	Eifriyt	demon	عفريت	فعليل	27.39.2

The noun of preeminence is used to express that one subject surpasses another in a certain quality expressed by the adjective used [81]. The two patterns that exist in this category are shown in Table 4.23.

Table 4.23: Triliteral Noun of Preeminence

Num.	Pattern	Stem	Gloss	Word	Pattern	QI
48	>CCaC	>Hsan	better	أحسن	أفعل	2.138.4
49	CuCCaY	AlmuvlaY	optimal	المثلى	فعلى	20.63.14

The Broken Plurals (BPs) are named as such because deriving the plural form from the singular is not a straightforward concatenative operation, where a plural suffix can be appended to the singular form. Rather, the singular form would be broken into segments and rearranged differently.

Table 4.24: A Singular Pattern with Different Broken Plural Patterns

Singular	Plural	Stem	QI	Gloss
sAHir	CaCaCap	saHarap	7.113.2	magician(s)
kAfir	CuCCAC	wAlkuf~Ar	9.68.5	non-believer(s)
SAHib	>CCAC	wa>SHAb	9.70.13	friend(s)
rAqid	CuCwC	ruqwd	18.18.4	sleeping person(s)
rAhib	CuCCAn	wAlruhbAn	9.34.9	priest(s)
sAjid	CuCCaC	suj~ad	12.100.7	prostrating person(s)
HAris	CaCaC	Haras	72.8.6	guard(s)
qA}im	CiCAC	qiyAm	39.68.21	standing person(s)
sAHil	CawACiC	sawAHil		coast(s)
qADy	CuCaCp	quDaAp		judge(s)
HAjj	CaCyC	hajyj		pilgrim(s)
hAlik	CaCCY	halkY		deceased person(s)

In her PhD thesis, Levy addressed the sound plurals and stated that this can be applied to the derived words, or words that are formed from other words by rules of morphological derivation (e.g., participles) [45]. McCarthy has also analysed this phenomenon very thoroughly and applied his prosodical theory on this type of plurals [51]. Neme also studied the transition from the singular to the broken plural through the traditional description that “The path from a singular form to a BP passes through a root.” [55].

There are no rules that can be generalised on the majority of patterns to form the broken plurals. Also, more than one pattern can be grouped under one broken plural pattern, or the opposite. To make this point clear, consider the words in Table 4.24, where the singular pattern CACiC, or Form I Active Participle, can lead to more than one Broken Plural.

Table 4.25: Popular Triliteral Broken Plural Patterns

Num.	Pattern	Stem	Gloss	Ar. Word	QI
50	CaCCY	marDY	patients	مرضى	4.43.23
51	>CCiCap	>jniHap	wings	أجنحة	35.1.10
52	CawACyC	qawAryr	bottles	قوارير	27.44.17
53	>CCaAC	>nhArA	rivers	أنهارا	71.12.9
54	CiCAC	AljibAl	moutains	الجبال	88.19.2
55	>CACYC	>HAdyv	talks	أحاديث	34.19.9
56	CuCwC	Albuywt	houses	البيوت	2.189.13
57	CuCuC	kutub	books	كتب	98.3.2
58	CwACiC	fawAkih	fruits	فواكه	23.19.10
59	CaCaCap	AlsaHarap	magicians	السّحرة	7.113.2
60	>CCuC	>$hur	months	أشهر	2.197.2
61	CaCACy	SayASyhm	castles	صياصيهم	33.26.8
62	CuC˜AC	Alz˜ur˜AE	farmers	الزّرّاع	48.29.41
63	>CACiC	Al>nAmil	fingers' tips	الأنامل	21.1.1
64	taCACyC	wtamAvyl	sculptures	وتماثيل	34.13.7
65	maCACyC	bmaSAbyH	glow-lamps	بمصابيح	67.5.5
66	CuCaCA'	AlfuqarA'	poor people	الفقراء	35.15.5
67	maCACiC	AlmaqAbir	cemeteries	المقابر	102.2.3
68	CaCA}iC	AlmadA}in	cities	المدآئن	7.111.6
69	CuCACY	sukArY	drunks	سكارى	4.43.9

The twelve singular stems in Table 4.24 share the same pattern CACiC, but their plurals are different, and that is why the Broken Plurals represent what is called the "non-concatenative" aspect of Arabic Morphology in its clearest form.

Table 4.25 shows the most common broken plural patterns in the lexicon. Two of the most common quadriliteral broken plural forms are shown in Table 4.26.

Table 4.26: Popular Quadriliteral Broken Plurals Patterns

Num.	Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
70	CaCACiC	AlHanAjir	throats	الحناجر	فعالل	33.10.13
71	CaCACyC	qarATys	papers	قراطيس	فعاليل	6.91.27

The pattern >CCAC is one of the most widely used broken plurals in the lexicon. The >CCAC pattern comes in the Quran as a plural pattern for two singular forms: namely CaCaC and CACiC (Form I Active Participle) as Table 4.27 shows. It is worth noting that the pattern CACiC was used with one root to form the singular stem of the plural >CCAC, which makes it safe to infer that >CCAC is not a preferred plural pattern for the Form I Active Participles.

Table 4.27: Singular Stems for the Broken Plural Pattern >CCAC

Singular	QI	Plural	QI	Gloss
SAHibkum	53.2.3	w>SHAb	9.70.13	friend-friends
AlbaSar	16.77.10	wAl>bSAr	10.31.10	eye sight-eye sights

Table 4.28 shows the root distribution according to their types. From numbers we see that the >CCAC pattern is distributed evenly between the strong and non-strong roots . There are 56 Strong roots that were used to form 139 surface words which occurred 427 times in the Quran. The surface form bears the definition of the surface word that was given in section **??**.

Table 4.28: The Roots Distribution over the >CCAC Pattern

Root Type	Occurrence	Surface Form	Roots
Strong	427	139	56
Geminate	51	20	13
Single Weak	338	116	35
Double Weak	154	34	7
Total	970	309	111

For the non-Strong roots that contain single weak, double weak, geminate, and all weak, 55 roots are used in the derivation of 170 surface forms, and they occurred 543 times. Although Strong roots are one more than non-Strong roots in numbers, 56 to be precise, the surface forms for the strong roots are fewer than those of non-strong roots by 31. The middle "w" category has the highest number of roots, 15, from which 54 surface forms are created, and they occurred 173 times. This is confirmed by Levy's finding [45].[9] Her conclusion was that for the pattern CaCC with the vowel"w" as the middle radical, >CCAC is the likely plural pattern, and our analysis supports that.

The >CCAC pattern employs 13 geminate roots to generate 20 surface forms that occurred 51 times. What is worth noting here is the appearance of both the middle and the last radical of the root in the generated stem. Consider the surface form ">sbAb" ("*reasons*") with the root "sbb", we notice the last two radicals are identical and they both appear on the surface Also the high number of geminate roots implies that >CCAC is somewhat the preferred plural pattern for geminate roots, and this is confirmed by Levy's finding.[10] She concluded that the geminate-rooted nouns employ >CCAC as the plural pattern more frequently than the CuCwC pattern.

4.4.2.4 Legal Combinations

We created a compatibility table whose 5,617 entries correspond to the legal prefix-pattern-suffix combinations, 2,052 from the Quran, and the rest from Modern Standard Arabic corpus represented by Alriyadh and Alnahar new papers. It is through this compatibility table that a root can be found. Since finding the correct pattern is a prerequisite step to finding the root, a pattern has to have an entry in the prefix-pattern-suffix table.

[9] See Table 2 on page 36 of Levy's PhD thesis.
[10] See Table 3 on page 38 of Levy's PhD thesis.

pattern : CwACiC (Broken Plural)
root : qfl fkh
stem : qwAfil fwAkih
"Caravans" *"fruits"*

pre	stem			suf	Allowed surface forms
∅	...	qwAfl fwAkh	...	k	qwAflk *"your caravans"* fwAkhk *"your fruits"*
b	...	qwAfl fwAkh	...	nA	bqwAflnA "by our *caravans*" bfwAkhnA *"by our fruits"*
...	...	...	...	...	...
wl	...	qwAfl fwAkh	...	y	wlqwAfly "and for *my caravans*" wlfwAkhy "and for *my fruits*"

Al	...	qwAfl fwAkh	...	∅	AlqwAfl *"the caravans"* AlfwAkh *"the fruits"*
bAl	...	qwAfl fwAkh	...	∅	bAlqwAfl "by the *caravans*" bAlfwAkh *"by the fruits"*
...	...	...	...	...	...
wll	...	qwAfl fwAkh	...	∅	wllqwAfl "and for *the caravans*" wllfwAkh "and for *the fruits*"

Figure 4.8: Prefix-Pattern-Suffix Table Example.

In Arabic Broken Plurals (BPs) Morphology, it is a rule that case-marking suffixes (An, At, wn, yn) cannot be attached to them, and only possessive pronouns (k, y, hm, etc.) are allowed to be appended to them on the surface. Furthermore, when the definiteness marker "Al" ("the") appears with a stem of this type, there should be no suffix of any type attached. Figure 4.8 depicts how this can be enforced using the compatibility table.

The words "qwAfl" ("caravans") and "fwAkh" ("fruits") share the same pattern CwACiC, but with different roots, "qfl" and "fkh" respectively. The allowed surface forms are those that either have possessive pronouns with no definiteness marker, or words with the definiteness marker, and its derivatives (bAl, fAl, kAl, wAl, ll, fbAl, fkAl, fll, wbAl, wkAl, wll), but with no suffixes.

Table 4.29 shows the most frequently used nominal prefix-suffix combinations in decreasing order. We notice that the nouns with no affixes constitute the majority of the lexicon entries. In the second row comes no prefix and the indefiniteness marker "A" as a suffix with 878 occurrences, followed by the definiteness determiner "Al" and no suffix with 753 occurrences.

Table 4.29: Nominal Prefix-Suffix Combinations

Prefix	Suffix	Freq.	Word	Ar. Word	QI
∅	∅	1108	|bA'	آباء	24.31.27
∅	A	878	>bkArA	أبكارا	56.36.2
Al	∅	753	Al|fAq	الآفاق	41.53.4
∅	p	358	|typ	آتية	20.15.3
∅	hum	204	|bA'hum	آباءهم	23.68.9
w	∅	189	w>dbAr	وأدبار	50.40.4
wAl	∅	185	wAl| SAl	والآصال	7.205.12
∅	h	169	>zwAjh	أزواجه	33.53.61
Al	yn	157	Al| vmyn	الآثمين	5.106.52
b	∅	152	b>smA'	بأسماء	2.31.11

4.4.3 Verbal System

The verbs in the Arabic lexicon constitute origins of words as many morphologists argue. Their reasoning goes back to the origin of life, where the first human had to do something (i.e. plant a land), and then later he/she called things around him/her names. Therefore the verb "Harava" ("*plow*") had taken place first, then the name for the tool with the pattern miCCAC "miHrAv" ("*plough*") was created [70].

4.4.3.1 Prefixes

Table 4.30: Verbal Popular Prefixes

Prefix	Freq.	Word	Ar. Word	QI
∅	2480	|tAk	آتاك	28.77.3
w	1014	w|ti	وآت	17.26.1
t	866	t>tnA	تأتنا	7.132.3
y	787	y>tyk	يأتيك	15.99.4
f	741	f|ti	فآت	30.38.1
l	233	l|twhA	لآتوها	33.14.9
n	196	n>ty	نأتي	13.41.4
w+y	154	wy>bY	ويأبى	9.32.7
l+y	108	ly>x*a	ليأخذ	12.76.16
>	96	>>sjdu	أأسجد	17.61.10

4.4.3.2 Suffixes

Table 4.31: Verbal Popular Suffixes

Suffix	Frequency	Word	Ar. Word	QI
∅	2173	|*ana	آذن	7.123.7
wA	996	|twA	آتوا	23.60.4
wn	670	f>rslwn	فأرسلون	12.45.11
t	361	>tat	آتت	18.33.3
nA	341	|tAnA	آتانا	9.75.6
hum	270	|tAhum	آتاهم	3.170.3
h	257	|tAh	آتاه	2.258.10
kum	224	|tAkum	آتاكم	5.48.40
tum	193	|tytum	آتيتم	2.233.55
n	165	>rDaEn	أرضعن	65.6.21

4.4.3.3 Patterns

Patterns in the Arabic morphology represent the containers, into which roots and short vowels are moulded to form the word. The verbal patterns are very small in number, 15 patterns, 12 of which are in common use. They are in the 3rd person.singular.masculine.perfective, or as the past form of a verb in English. Table 4.32 shows the 10 most popular triliteral verbal patterns in Arabic [81].

Table 4.32: Triliteral Verbal Patterns

Num.	Pattern	Stem	Gloss	Ar. Word	QI
1	CaCaCa	kataba	wrote	كتب	6.54.9
2	CaC~aCa	bad~lnA	changed	بدّلنا	76.28.7
3	CACaCa	fHAsabnAhA	accounted for	فحاسبناها	65.8.9
4	>CCaCa	>slama	became muslim	أسلم	2.112.3
5	taCaC~aCa	tataqal~b	turned into	تتقلّب	24.37.16
6	taCaACaCa	ta$Abahat	look alike	تشابهت	2.118.18
7	AnCaCaCa	Anqalaba	returned	انقلب	22.11.16
8	AiCtaCaCa	Aiqtaraba	got closer	اقترب	21.1.1
9	AiCCaCCa	AibyaD~at	white	ابيضّت	3.107.3
10	AistaCCaCa	Aistakbara	arrogant	استكبر	38.74.3

For the quadriliteral verbs, there are three patterns, two of which occurred in the Quran, and are very common in everyday language. Table 4.33 shows these two patterns.

Table 4.33: Common Quadriliteral Verbal Patterns

Num.	Pattern	Stem	Gloss	Ar. Word	Ar. Pat.	QI
11	CaCCaCa	HaSHaSa	got clear	حصحص	فعلل	12.51.21
12	ACCaCC~a	ATma>nantum	felt ease	اطمأننتم	افعللّ	4.103.11

4.4.3.4 Legal Combinations

In Table 4.34, it can be seen that the verbs without any affixes are the most common, followed by the conjunction particle "w" with no suffix, with 322 as its frequency. This validates the results of the SEQUITUR algorithm. In the third row, verbs without a prefix and the subject pronoun "wA" (masculine.plural.nominative) as a suffix has appeared 310 times in the lexicon. Furthermore, the suffix "wn" (masculine.plural.nominative) with the prefixes "t, y" in the 9th and the 10th row are good indicators for identifying imperfective verbs.

Table 4.34: Verbal Prefix-Suffix Combinations

<table>
<tr><th>Prefix</th><th>Suffix</th><th>Frequency</th><th>Word</th><th>Ar. Word</th><th>QI</th></tr>
<tr><td>∅</td><td>∅</td><td>614</td><td>|*n</td><td>آذن</td><td>7.123.7</td></tr>
<tr><td>w</td><td>∅</td><td>322</td><td>w|t</td><td>وآت</td><td>17.26.1</td></tr>
<tr><td>∅</td><td>wA</td><td>310</td><td>|twA</td><td>آآتوا</td><td>23.60.4</td></tr>
<tr><td>t</td><td>∅</td><td>268</td><td>t>s</td><td>تأس</td><td>5.26.11</td></tr>
<tr><td>y</td><td>∅</td><td>258</td><td>y>x*</td><td>يأخذ</td><td>18.79.14</td></tr>
<tr><td>∅</td><td>t</td><td>244</td><td>|tt</td><td>آتت</td><td>18.33.3</td></tr>
<tr><td>f</td><td>∅</td><td>231</td><td>f|t</td><td>فآت</td><td>30.38.1</td></tr>
<tr><td>w</td><td>wA</td><td>204</td><td>w|twA</td><td>وآتوا</td><td>2.43.3</td></tr>
<tr><td>y</td><td>wn</td><td>189</td><td>y>x*wn</td><td>يأخذون</td><td>7.169.7</td></tr>
<tr><td>t</td><td>wn</td><td>183</td><td>t>kulwn</td><td>تأكلون</td><td>3.49.33</td></tr>
</table>

4.5 SAS Implementation Overview

Although SAS is composed of three main functions; "segment", "analyze", and "extract", there are a number of data structures and supporting functions that have to be detailed. The SAS implementation explained hereafter includes Java classes from the Lucene search engine as well as Java supporting functions that have been developed by the author.

4.5.1 SAS Major Classes and Functions

The first major Class is the lexicon, which includes the UTF-8 sorted Arabic characters array. The first step undertaken by SAS is to map every character in the input word to its index according to its position within the character's array. The Lexicon Class also contains the major lexical components including prefixes, suffixes, patterns, roots, and legal prefix-pattern-suffix combinations. It is worth mentioning that when an illegal prefix-suffix combination is considered, a “null” value is assigned to that combination’s position within the “Combos” array. For instance, the preposition prefix “b” and the suffix “A” can not be combined into a single word, and that is indicated by assigning the “null” value to the combination slot in the “Combos” array as shown in the code labelled Java Code 4.1.

Java Code 4.1: SAS Lexicon Class

```
public class SASLexicon {
        public final char[ ] UTFArabic = new char [ ] {
            '\u0621', '\u0622', '\u0623', '\u0624', '\u0625', '\u0626', '\u0627',
            '\u0628', '\u0629', '\u062a', '\u062b', '\u062c', '\u062d', '\u062e',
            '\u062f', '\u0630', '\u0631', '\u0632', '\u0633', '\u0634', '\u0635',
            '\u0636', '\u0637', '\u0638', '\u0639', '\u063a', '\u0641', '\u0642',
            '\u0643', '\u0644', '\u0645', '\u0646', '\u0647', '\u0648', '\u0649',
            '\u064a', '\u064b', '\u064c', '\u064d', '\u064e', '\u064f', '\u0650',
            '\u0651', '\u0652'
        };
        public final int[ ][ ] Prefixes = new int[ ][ ]{
            {}, // “empty prefix”
            {6, 29}, // the definiteness marker “Al” (ال)
            {7}, // the preposition letter “b” (ب)
            ...
        };
        public final int[ ][ ] Suffixes = new int[ ][ ]{
            {}, // “empty suffix”
            {6}, // the suffix “A” (ا)
            {6, 9}, // the plural feminine marker “At” (ات)
            ...
            {33, 31} // the plural masculine marker “wn” (ون)
        };
        public final int[ ][ ] Patterns = new int[ ][ ]{
            {-1, -2, -3}, // the pattern “CCC” (فعل)
            {-1, -2, 6, -3, 35, -4}, // the quadriliteral pattern “CCACYC” (فعاليل)
            {-1, 6, -2, -3}, // the pattern “CACC” (فاعل)
            ...
```

Java Code 4.1: SAS Lexicon Class – continued

```
        {30, 31, -1, -2, -3} // the pattern "mnCCC" (منفعل)
    };
    public final int[ ][ ] Roots = new int[ ][ ]{
        {2, 7, 7}, // the root ">bb" (أبّ)
        ...
        {35, 33, 30} // the root "ywm" (يوم)
    };
    public final int[ ][ ][ ] Combos = new int[ ][ ][ ]{
        {
            {0, 1, ..., 98}, // This is the combination (∅−∅)
            {0, 1, ..., 98}, //(∅−A)
            ...
        },
        ...
        {
            {0, 1, ..., 98}, //(b−∅)
            null, //(b−A)
            ...
        },
        {
            {0, 1, 2, 3, 4, 13, 17, 18, 51, 55}, //(wll−∅)
            null, //(wll−A)
            {4, 51, 52}, //(wll−At)
            ...
        }
    };
}
```

The next major data structure is the "Template" Class, which is shown in Java Code 4.2. This Class represents the input word's morphological units; including the prefix, the stem, the suffix, the root, and the pattern.

Java Code 4.2: SAS "Template" Class

```
public class Template{
        int [ ] stem;
        int [ ] root;
        int prefix, suffix, pattern;
        public Template(int iprefix, int [ ] istem, int isuffix, int [ ] iroot, int ipattern){
                prefix = iprefix;
                stem = istem;
                suffix = isuffix;
                root = iroot;
                pattern = ipattern;
        }
}
```

Having developed SAS this way helped in speeding up the processing time because all arrays are sorted and searched using the binary-search algorithm. Since the root's radicals are indicated by a negative integer in the pattern template as Table 4.3 shows, the extraction of a radical is done by fetching the character in the stem array that corresponds to the negative value within the pattern array as shown in Java Code 4.3.

Java Code 4.3: SAS "extract" Function

Input: A stem and a pattern representing an Arabic word.
Output: An array of integers representing the root's radicals.

```
public int[ ] extract(int[ ] stem, int[ ] pattern){
        int[ ] root = new int[5];
        int radicals = 0;
        for (int x = 0; x < pattern.length; x++)
                if (pattern[x] < 0)
                        root[radicals++] = stem[x];
        return java.util.Arrays.copyOf(root, radicals);
}
```

The input word has to have a valid pattern that exists in the "Patterns" array. The function "matchingPattern" whose code is given in Java Code 4.4 matches the stem array to the pattern by comparing the positive values in the pattern array to the values that occupy the same slots in the stem array, if they are equal, then it returns true. If any of the positive values in the pattern array happens not to match its corresponding value within the stem array, the function returns false.

Java Code 4.4: SAS "matchingPattern" Function

Input: Two integer arrays representing the stem and the pattern of the input word.

Output: True if the pattern matches the stem, false otherwise.

```
public boolean matchingPattern(int [ ] stem, int [ ] pattern){
        for(int x = 0; x < pattern.length; x++)
            if(pattern[x] >= 0 && pattern[x] != stem[x])
                return false;
        return true;
}
```

The "segment" function starts with a stem with no prefix and no suffix as shown in Java Code 4.5, then a stem with a single-character prefix and so on until all possible combinations are produced. Since the language alphabets are represented as integers, prefixes and suffixes are represented by a sorted two-dimensional arrays, and they are always referenced by their array indices, the same is followed with patterns.

Java Code 4.5: SAS "segment" Function

Input: An integer array representing the input word.

Output: An array of all possible prefix-stem-suffix combinations for the input word.

```
public Template[ ] segment(int[ ] word){
        Template[ ] segments = new Template[25];
        int[ ] prefix, stem, suffix;
        int p_index , s_index;
        int count = 0;
        for (int prefix_len = 0; prefix_len <= 5 && prefix_len <= word.length; pre-
```

Java Code 4.5: SAS "segment" Function – continued

```
fix_len++) {
            prefix = new int[prefix_len];
            System.arraycopy(word, 0, prefix, 0, prefix_len);
            int stem_len = (word.length - prefix_len);
            int suffix_len = 0;
            for (; stem_len >= 1 && suffix_len <= 6;) {
                stem = new int[stem_len];
                System.arraycopy(word, prefix_len, stem, 0, stem_len);
                suffix = new int[suffix_len];
                System.arraycopy(word, prefix_len + stem_len, suffix, 0, suffix_len);
                p_index = binarySearch(Prefixes, prefix);
                s_index = binarySearch(Suffixes, suffix);
                if (p_index >= 0 && s_index >= 0)
                    segments[count++] = new Template(p_index, stem, s_index, null, -1);
                stem_len--;
                suffix_len++;
            }
        }
        return (count > 0) ? java.util.Arrays.copyOf(segments, count) : null;
}
```

The "analyze" function is shown in Java Code 4.6. It validates the morphological combination of a given word by checking the "Combos" array for the existence of that particular pattern.

Java Code 4.6: SAS "analyze" Function

Input: An array of all possible prefix-stem-suffix combinations for the input word.

Java Code 4.6: SAS "analyze" Function – continued

Output: An array of only the **valid** prefix-pattern-suffix combinations.

```
public Template[ ] analyze(Template[ ] segments){
    Template[ ] valids = new Template[25];
    int count = 0;
    for (Template segment : segments)
        for(int x = 0; x < Patterns.length; x++)
            if (matchingPattern(segment.stem, Patterns[x]))
                if (binarySearch(Combos[segment.prefix][segment.suffix], x) >= 0){
                    int [ ] root = extract(segment.stem, Patterns[x]);
                    valids[count++] = new Template(segment.prefix, segment.stem, seg-
ment.suffix, root, x);
                }
    return (count > 0) ? java.util.Arrays.copyOf(valids, count) : null;
}
```

It goes directly to the [prefix][suffix] slot within the "Combos" array to look for the pattern. The combination is valid if the the binarySearch function returns an integer that is greater than -1.

4.5.2 Integrating SAS into Lucene

SAS has been integrated into Lucene using three Lucene Classes. The SASAnalyzer class shown in Java Code 4.9 utilises two built-in classes-the TokenFilter and the CharTokenizer. The SASTokenizer Class extends the built-in Class CharTokenizer, and is shown in Java Code 4.7. The main functionality of this class is to make sure that the input characters are acceptable for the analyzer. So the only over-ridden function in the class is a boolean function named "isTokenChar", which takes in an input character and returns true if the character is a UTF-8 Arabic character by searching the "UTFArabic" array for that character using the binarySearch algorithm.

Java Code 4.7: SASTokenizer Class

```
import org.apache.lucene.analysis.CharTokenizer;
import org.apache.lucene.util.Version;
public class SASTokenizer extends CharTokenizer {
        public SasTokenizer(java.io.Reader input) {
                super(Version.LUCENE_35, input);
        }
        protected boolean isTokenChar(int c) {
                return (java.util.Arrays.binarySearch(UTFArabic, (char) c) >= 0);
        }
}
```

The following class Java Code 4.8 is responsible for replacing each word in the input stream with its analysed form; either a stem or a root depending on the indexing choice. The built-in function “incrementToken” has been over-ridden to process the input token using the SAS methods. It ought to be observed that the class “CharTermAttribute” retains all information pertaining to the currently processed token, which can be in turn modified and updated accordingly. This class stores vital data that can be of great use for our solution; for instance it contains the buffer holding the token’s characters that will be used for the indexing and searching. We use this class to store the SAS stems and roots in place of the input raw words so that the indexing can be undertaken using the selected term.

Java Code 4.8: SASFilter Class

```
import org.apache.lucene.analysis.TokenFilter;
```

Java Code 4.8: SASFilter Class – continued

```
import org.apache.lucene.analysis.TokenStream;
import org.apache.lucene.analysis.tokenattributes.CharTermAttribute;
public class SASFilter extends TokenFilter {
    private CharTermAttribute termAttribute = null;
    private int whichStemmer = -1;
    private char[ ] stem = null;
    private char[ ] root = null;
    public SASFilter(TokenStream input, int w) throws FileNotFoundException{
        super(input);
        termAttribute = addAttribute(CharTermAttribute.class);
        whichStemmer = w;
    }
    public boolean incrementToken() throws IOException {
        if (input.incrementToken())
            if(whichStemmer == 0){
                root = getRoot(termAttribute.buffer());
                termAttribute.copyBuffer(root, 0, root.length);
                return true;
            }
            else{
                stem = getStem(termAttribute.buffer());
                termAttribute.copyBuffer(stem, 0, stem.length);
                return true;
            }
        return false;
    }
}
```

The “Analyzer” class employs the previous two classes by first declaring a variable named “result” of type “CharTokenizer” to make certain all characters in the “input” stream are validated to be UTF-8 Arabic letters. The parameter “field” refers to the name of the field whose contents are stored in the parameter "reader", according to the TREC collection, these fields are: the "title", the "description" , and the "text" or the document's body itself.

Java Code 4.9: SASAnalyzer Class

```
import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.TokenStream;
import org.apache.lucene.analysis.StopFilter;
import org.apache.lucene.util.Version;
public final class SasAnalyzer extends Analyzer{
        private Version matchVersion;
        private int which;

        public SasAnalyzer(Version mv, int w) {
            matchVersion = mv;
            which = w;
        }
        public final TokenStream tokenStream(String field, java.io.Reader reader) {
            TokenStream result = new SASTokenizer(reader);
            try{
            result = new StopFilter(matchVersion, result, StopFilter.makeStopSet(matchVersion,
getStopWords()));
            } catch (java.io.IOException e) {
                System.out.println(e.getMessage());
                System.exit(-1);
            }
            try{
                result = new SASFilter(result, which);
            } catch (java.io.FileNotFoundException e) {
                System.out.println(e.getMessage());
                System.exit(-1);
            }
            return result;
        }
}
```

The second operation is to remove the stop words from the "result" stream, leaving only words that have to be stemmed. We utilised Khoja's list of stop words for our solution. The final step is to filter this stream by instantiating a "SASFilter" class whose main function is to substitute each word in the input stream with its stemmed form, depending on the integer parameter "which", 0 for the SAS root, 1 for the SAS stem, 2 for Khoja root, and so on.

4.6 Conclusion

We proposed a concise morphological lexicon, which is based on the Quran morphology in order to enhance the Arabic Information Retrieval performance. Using the Quran corpus, we have inferred 82 nominal and 12 verbal patterns. This is leveraged in the word segmentation module so that affix removal is carried out according to the Quran rules in order to learn the morphological borders amongst different prefix-pattern-suffix combinations. We have also presented the Java code necessary to integrate our SAS into the Lucene search engine.

Chapter 5

Evaluation

A great deal of research has been conducted on assessing the impact the indexing term would have on AIR performance. And yet the Arabic language still lacks NLP tools aimed at resolving word ambiguity due mainly to the complexity of the Arabic morphology. We will explore the previous AIR experiments aimed at appraising which form of the word can lead to optimal AIR performance, and present their findings.

The previous Arabic stemming approaches fall into two main classes; root or light stemming. A **light** stemmer stops processing after removing frequent prefixes and suffixes from the input word. The output of the light stemming process is the stem; which bears within it the root and the morphological pattern. The **root** stemmer has to go one further step, which is to find out the pattern of the stem and extract the root radicals.

Using two tests collections, the TREC and the ZAD datasets, our proposed SAS root method is compared against Khoja and Sebawai stemmers. Moreover, the proposed SAS stem method is evaluated against light10, Al-Stem, MADA, and AMIRA SVM. The lessons benefited from running these experiments are discussed along with the issues facing future Arabic morphological analysis research.

5.1 Standard IR Evaluation Methods

The standard approach to information retrieval system evaluation revolves around the notion of relevant and non-relevant documents [49]. However, in order to reach a satisfying level of confidence in the results returned by the IRS, the following items have to be at hand:

- A standard document collection
- A suite of user queries
- A set of relevance judgments, a binary assessment of either relevant or non-relevant for each query-document pair assigned by a human judge.
- Common measures that can evaluate the quality of a list of retrieved documents.

In the next two sections, we go over the test collections used in our experiments, and the standard measures employed in gauging the effectiveness of our SAS method in comparison to the other stemming methods.

5.2 TREC Collection

The introduction of the Arabic language to the Cross-Language Information Retrieval (CLIR) track in 2001 during the tenth Text REtrieval Conference (TREC-2001) has offered researchers a set of documents, queries, and relevant judgments to evaluate their ideas on a standard test collection [29].

The TREC-2002 corpus has enabled researchers to resolve issues in Arabic morphology without worrying about the evaluation data. Prior to the existence of the TREC corpus, every solution had its own unique testing data. The collection has also enabled research institutes, such as University of California (UC) at Berkeley, to create a concordance needed to derive light stemmers based on n-gram frequencies.

5.2.1 TREC Documents

The collaboration effort between the Linguistic Data Consortium (LDC), the National Institute of Standards and Technology (NIST), and TREC-2001/TREC-2002 participants had produced 383,872 tagged documents. The 383,872 Arabic articles collected from the Agence France Press (AFP), covering the period from 1994 to 2000, had been prepared by Linguistic Data Consortium (LDC) to enable researchers evaluate their new Information Retrieval ideas.

As Figure 5.1 shows, these articles were written in Modern Standard Arabic (MSA) with no short vowels, and each document was broken into different fields tagged using TIPSTER-style SGML, and transcoded to Unicode (UTF-8). The segment labelled "HEADLINE" is the title of the article, whereas "TEXT" segment containing the content of the article is broken into paragraphs.

```
<DOC>
<DOCNO>20001208_AFP_ARB.0003</DOCNO>
<HEADER>ارا7100 2 ش 7200 قبر/ افب-اهخ84 ملاحظة</HEADER>
<BODY>
<HEADLINE>الى السادة المشتركين</HEADLINE>
<TEXT>
<P>
    نيقوسيا 21-8 (اف ب) - تعتذر وكالة فرانس برس عن انقطاع البث لاسباب فنية طارئة خارجة عن
    ارادتها وتستأنف الارسال.
 </P>
</TEXT>
<FOOTER>______ غ ر افب</FOOTER>
</BODY>
<TRAILER>جمت ديس 00 406080</TRAILER>
</DOC>
```

Figure 5.1: TREC Arabic Document Example.

In this example, the document consists of one paragraph saying "*AFP apologises about the interruption of the broadcast due to technical reasons that are out of its reach, and it would resume transmitting*".

5.2.2 TREC Topics or Queries

The queries, or topics as they are referred to by TREC participants, had been developed through a collaboration effort between the National Institute of Standards and Technology (NIST), the Linguistic Data Consortium (LDC), and the Text REtrieval Conference (TREC) participants [29].

<top>

<num> Number: AR7
<title>
Criticism and political poetry in the Arab World

<desc> Description:
How have Arab critics expressed their attitude towards the new political poetry, whether for or against the political regime in their countries?

<narr> Narrative:
Stories about festivals of poetry in the Arab World may be related to this topic. Any article concerning politics or arts that does not mention literary criticism is not relevant to this topic.

</top>

<top>

<num> Number: AR7
<title>
النقد والشعر السياسي في العالم العربي

<desc> Description:
كيف يعبر النقاد العرب عن مواقفهم تجاه الشعر السياسي سواء كان مع او ضد النظام السياسي في بلدهم؟

<narr> Narrative:
يمكن ارفاق الاخبار المتعلقة بالمهرجانات الشعرية في العالم العربي بالموضوع و لكن الاخبار السياسية والاخبار الفنية الخارجة عن ميدان النقد الادبي لا علاقة لها بالموضوع.

</top>

Figure 5.2: TREC English/Arabic Topic Example.

Figure 5.2 shows a sample of a topic document with title, description and narrative fields. The "title" is a short statement describing the subject of the topic, whereas the "description" is a question-like statement enquiring about a certain subject. The "narrative" explains briefly which subjects are considered relevant to the topic, and it is intended to assist human judges to weight the relevance of each document.[2]

Twenty five English queries were introduced in TREC-2001, which were then translated to other languages, including Arabic, in an attempt to conduct Cross Language Information Retrieval (CLIR) experiments. Another fifty queries were submitted during TREC-2002 proceedings to make seventy five queries or topics [59].

5.2.3 TREC Relevance Judgments

Ten teams evaluated their AIR systems in TREC-2001 using twenty five queries, where they were asked to find relevant documents written in Arabic related to queries written in English, and return a ranked list of the top 1,000 relevant documents. They were also required to submit at least one run using only the title and the description field of the topic. The ranked documents were then submitted to human judges for manual examination to decide the relevance (YES/NO) for each document in the pool. The objective was to develop a *gold standard* for the tests collection.

In TREC-2002, nine participants performed the same task on the other 50 topics, which had been previously translated by LDC to Arabic. The participants from both conferences were able to manually judge 10,031 documents, where each document is given a value of 1 or 0 indicating its relevance to the query. The mean number of relevant documents for a query was 165.

5.3 The ZAD Collection

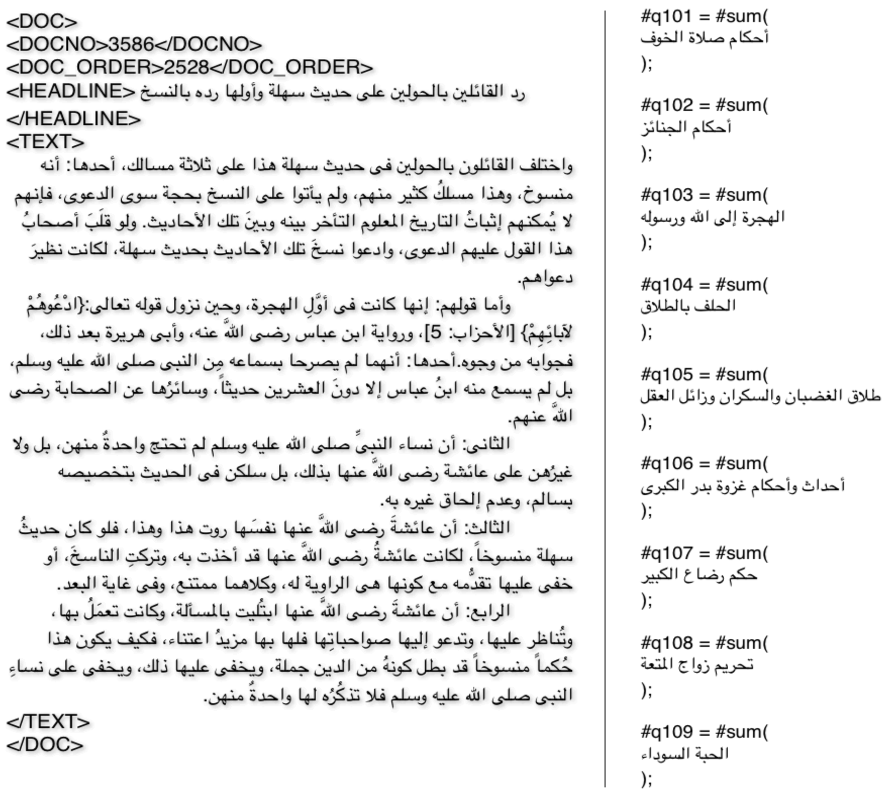
As it was stated in Section 5.2.1, the TREC tests collection is written in Modern Standard Arabic (MSA). Therefore, there was a need for measuring the effectiveness of the different stemming methods over the Classical Arabic Language. Darwish had developed such a collection as part of his PhD studies [20]. It is

[2] Figure 5.2 shows exact copies of topic number 7 taken from the English/Arabic Topics collection.

based on an electronic copy of a traditional 14th century book titled zAd AlmEAd (زاد المعاد في هدي خير العباد) [46].

5.3.1 ZAD Documents, Queries, Relevance Judgments

The ZAD collection consists of 2,730 documents representing the Classical Arabic Language, 25 queries, and originally 530 manually-judged relevant documents prepared by Darwish [20].



```
<DOC>
<DOCNO>3586</DOCNO>
<DOC_ORDER>2528</DOC_ORDER>
<HEADLINE> رد القائلين بالحولين على حديث سهلة وأولها رده بالنسخ
</HEADLINE>
<TEXT>
واختلف القائلون بالحولين فى حديث سهلة هذا على ثلاثة مسالك، أحدها: أنه
منسوخ، وهذا مسلكُ كثير منهم، ولم يأتوا على النسخ بحجة سوى الدعوى، فإنهم
لا يُمكنهم إثباتُ التاريخ المعلوم التأخر بينه وبينَ تلك الأحاديث. ولو قلَبَ أصحابُ
هذا القول عليهم الدعوى، وادعوا نسخَ تلك الأحاديث بحديث سهلة، لكانت نظيرَ
دعواهم.
وأما قولهم: إنها كانت فى أوَّلِ الهجرة، وحين نزول قوله تعالى:{ادْعُوهُمْ
لآبَائِهِمْ} [الأحزاب: 5]، ورواية ابن عباس رضى اللهّ عنه، وأبى هريرة بعد ذلك،
فجوابه من وجوه.أحدها: أنهما لم يصرحا بسماعه مِن النبى صلى الله عليه وسلم،
بل لم يسمع منه ابنُ عباس إلا دونَ العشرين حديثاً، وسائرُها عن الصحابة رضى
اللهّ عنهم.
الثانى: أن نساء النبىِّ صلى الله عليه وسلم لم تحتج واحدةٌ منهن، بل ولا
غيرُهن على عائشة رضى اللهّ عنها بذلك، بل سلكن فى الحديث بتخصيصه
بسالم، وعدم إلحاق غيره به.
الثالث: أن عائشةَ رضى اللهّ عنها نفسَها روت هذا وهذا، فلو كان حديثُ
سهلة منسوخاً، لكانت عائشةُ رضى اللهّ عنها قد أخذت به، وتركتِ الناسخَ، أو
خفى عليها تقدُّمه مع كونها هى الراوية له، وكلاهما ممتنع، وفى غاية البعد.
الرابع: أن عائشةَ رضى اللهّ عنها ابتُليت بالمسألة، وكانت تعمَلُ بها،
وتُناظر عليها، وتدعو إليها صواحباتِها فلها بها مزيدُ اعتناء، فكيف يكون هذا
حُكماً منسوخاً قد بطل كونهُ من الدين جملة، ويخفى عليها ذلك، ويخفى على نساءِ
النبى صلى الله عليه وسلم فلا تذكُرُه لها واحدةٌ منهن.
</TEXT>
</DOC>
```

```
#q101 = #sum(
أحكام صلاة الخوف
);

#q102 = #sum(
أحكام الجنائز
);

#q103 = #sum(
الهجرة إلى الله ورسوله
);

#q104 = #sum(
الحلف بالطلاق
);

#q105 = #sum(
طلاق الغضبان والسكران وزائل العقل
);

#q106 = #sum(
أحداث وأحكام غزوة بدر الكبرى
);

#q107 = #sum(
حكم رضاع الكبير
);

#q108 = #sum(
تحريم زواج المتعة
);

#q109 = #sum(
الحبة السوداء
);
```

Figure 5.3: ZAD Document and Query Sample.

I communicated with the author regarding some discrepancies that appeared in the relevance files, which pertain to the presence of some document numbers that were not part of the actual corpus documents. This correspondence resulted in eliminating 83 documents from the relevance file, leaving only 447 relevant documents. Furthermore, query number "q123" has no relevant documents, and it has been deleted from the queries list thus.

Figure 5.3 shows a sample document and the first 8 queries respectively. As it can be noticed, the queries are composed of few words, mostly two words. There is a word that is repeated in more than one query, and had impacted the precision of the stemmers because it is deemed a stop word; namely the word "AHkAm" or "Hkm" (أحكام، حكم) in queries q101 and q102 in Figure 5.3. After removing this word, q101 becomes "SalAp Alxwf" (صلاة الخوف), and q102 becomes "AljnA}z" (الجنائز), and this has improved SAS root results by 3%.

There are also some documents that were judged to be relevant even they have no common terms with the query. Query number q102 for instance talks about funerals (أحكام الجنائز), and document number 2877 is judged to be relevant but the document is about crying and mourning the dead and does not mention funerals.

5.4 Common Evaluation Measures

There are two values that are employed in evaluating an Information Retrieval System (IRS): precision and recall. The precision (Equation 5.1) is calculated by dividing the number of relevant documents retrieved by the total number of retrieved documents. It is used a measure the ability of the IRS to present only relevant documents.

$$Precision = \frac{Relevant\ Documents\ Retrieved}{Retrieved\ Documents} \tag{5.1}$$

The recall on the other hand is the ratio of relevant documents retrieved over the total number of relevant documents in the collection (Equation 5.2). It is a measure of the ability of the IRS to present all relevant documents.

$$Recall = \frac{Relevant\ Documents\ Retrieved}{Total Relevant\ Documents} \tag{5.2}$$

The optimal scenario occurs when the retrieval system records a 100% precision, which indicates that all retrieved documents are relevant. Similarly, a 100% recall means that all relevant documents are retrieved.

The Precision/Recall measures evaluate the quality of an unordered set of retrieved documents [15]. But in order to evaluate the quality of an ordered, or *ranked*, lists of retrieved documents, precision can be plotted against recall after each retrieved document. To facilitate computing average performance over a set of topics, each with a different number of relevant documents, individual topic precision values are interpolated to a set of standard recall levels (0 to 1 in increments of 0.1) [79].

The precision computed after a given number of documents have been retrieved reflects the actual measured system performance as a user might see it. Each document precision average is computed by summing the precisions at the specified document cutoff value and dividing by the number of topics. Our aim is walk through the steps that any web user might take, it is common to assume that web information seekers do not usually read more than the first 20 documents, therefore, we present the resulting precision at 5, 10 , 15, and 20 document cutoff values.

Mean average precision (MAP) is the single-valued summary measure used when an entire graph is too cumbersome. The average precision for a single topic is the mean of the precision obtained after each relevant document is retrieved (using zero as the precision for relevant documents that are not retrieved). The mean average precision for a run consisting of multiple topics is the mean of the average precision scores of each of the individual topics in the run.

5.5 Work Evaluating AIR Prior to TREC

One of the first recorded attempts to tackle the Arabic issue in IR was conducted in 1991 at the Illinois Institute of Technology (IIT) by Ibrahim Al-Kharashi under the supervision of Martha Evens [2]. They developed a micro Arabic Information Retrieval System (Micro-AIRS) that was aimed at studying the impact of stemming methods on retrieval performance. Al-Kharashi obtained a data set for the evaluation from the King Abdulaziz City for Science and Technology (KACST) database, which included 355 documents in the field of Computer and Information Science [9].

Al-Kharashi built a list of 3,442 keywords, from which 1,126 were chosen to be part of the lexicon. The lexicon contained 725 stems, 526 roots, and all the stop words encountered during the data processing phase. This lexicon was necessary to avoid the development of complicated linguistic stemming and root finding algorithms. These stems and roots were the same ones that were accessed during the retrieval process. For relevance judgements, Al-Kharashi created 50 queries and divided the 355 documents into three sets. Each set was manually judged by a Computer Science student. At the end, only 29 queries were chosen because they had one or more relevant documents. There were 198 manually-judged documents in the collection.

The conclusion was that using the root as an indexing term outperformed the surface word and the stem methods in retrieving more relevant documents, but with the downside that more irrelevant documents were retrieved as well. Table 5.1 shows the average retrieval results they achieved for the 50 queries they prepared for the experiment.[1]

[1] This is reproduced from Table 7 on page 62 of Al-Kharashi's PhD thesis [9].

Table 5.1: Micro-AIRS Experimental Results

Unit	Retrieved	Relevant	Irrelevant
surface word	2.24	2.03	0.21
stem	7.79	3.69	4.10
root	12.55	4.72	7.83

Even though the data and the lexicon are not comprehensive, the experiment still shows that more terms are conflated under the root even if their meanings are not related. In other words, the root conflates more words under one indexing term without taking the semantic relationship into account. Al-Kharashi's conclusion that the root was more efficient as an indexing term than the stem didn't hold when a larger corpus was made available; TREC experiments showed that the stem gave better performance when used as an indexing term than the root [6; 44; 58].

5.6 TREC Experiments

The TREC-2001 and TREC-2002 had led to making several stemming methods available to the research community including light10 [44], and Sebawai [19]. The University of California (UC) at Berkeley conducted research on AIR, supported by the Defence Advanced Research Projects Agency (DARPA), and they created two stemmers in the process, one of them is a light stemmer [18].

At TREC-2001, the University of California submitted an experiment to the Cross-Language IR track with a simpler version of their light stemmer that removes the definite articles from nouns and four suffixes (An, wn, At, p). Then they modified their stemmer based on another experiment they ran on the same document collection to calculate the frequencies of prefixes and suffixes.

The idea is to create six lists, three lists for prefixes and three for suffixes. Each prefix list contains the initial, the first two, the first three characters of Arabic words respectively. The same is done with the final, the last two and the last three characters for the lists of suffixes, respectively. They then sorted the lists according to their frequencies in descending order. Table 5.2 shows their most frequent prefixes in TREC-2002 Arabic corpus; the same procedure was undertaken to determine the same lists for suffixes.

Table 5.2: The most Frequent Corpus Prefixes Found By UC

1-char		2-char		3-char	
prefix	frequency	prefix	frequency	prefix	frequency
w	117324	Al	55364	wAl	19411
A	94043	wA	32787	bAl	12711
b	49319	bA	16789	fAl	9079

They conducted an AIR experiment where they ran their light stemmer against Al-Stem. What is interesting about their run is that their light stemmer outperformed Al-Stem, as Table 5.3 shows. This is no surprise given the results of our experiments, we will show in section 5.8 how Al-Stem's patterns degrade its performance. Al-Stem was developed based on a probabilistic model, so the frequency of the word was the only factor in selecting its pattern as part of the lexicon. Therefore, some of these patterns turned out to be not according to Arabic morphological rules.

Table 5.3: Al-Stem vs. the University of California Light Stemmer

		without expansion		with expansion	
run id	stemmer	recall	precision	recall	precision
mon0	NONE	4035	0.2365	4583	0.2872
mon3	Al-Stem	4500	0.2858	4864	0.3482
BKYMON	Berkeley LS	4543	0.3099	4952	0.3666

The run named "mon0" was produced with no stemming involved, and 40 trigrams were selected from the top-ranked 10 documents as terms for the query expansion. The "mon3" and "BKYMON" runs were produced using the Al-Stem and and the University of California light stemmer respectively, and 20 words were selected from the top-ranked 10 documents for the query expansion.

Another experiment was conducted by University of Massachusetts to evaluate the difference between a root and a stem as indexing terms [43; 44]. A clustering-algorithm based on removing vowels from Arabic words was implemented using co-occurrence analysis. This produced stem classes that were better than no stemming, but inferior to good light stemming or morphological analysis. The team reported that the light stemmer was more effective for Cross-Language Information Retrieval (CLIR) than a morphological stemmer that tried to find the root of the word. They used Khoja stemmer to find the root.

Aljlayl had conducted an experiment measuring the effectiveness of the root and the stem as indexing terms, and he reported that the stem achieved a 19.6% improvement over the root as Table 5.4 shows [6; 7].

Table 5.4: Mean Average Precision

Index	MAP
Surface Word	0.1979
Root	0.2987
Light Stem	0.3715

5.7 The Evaluation of the Root as an Indexing Term

Using the procedure followed by the TREC participants, the results reported here were carried out using the topic's title and description fields as queries. TREC participants used the top 15 terms from the top 10 documents for measuring the relevance feedback impact. We did not go measure the relevance feedback factor because it offers no new insights or knowledge to the stemming issue which we try to tackle in this study.

5.7.1 Root vs. Stem

An experiment was conducted to assess the difference between the root and the stem as indexing terms. SAS was used in the experiments to generate both indexing terms; the root and the stem. For stop words, we used Khoja's list for SAS or any stemmer that has no stop words list (e.g., MADA). As for the other stemmers, Sebawai and Al-Stem have their own stop words list, and light10 uses Khoja's list. The search engine we used was Lucene from Apache.[3]

For the TREC collection, Table 5.5 shows the mean average precisions for the three indices; the raw surface word, the root, and the stem respectively.

Table 5.5: Mean Average Precision

Index	MAP
No Stemming	0.1949
SAS Root	0.2538
SAS Stem	0.3035

For the ZAD collection, Table 5.6 shows the mean average precision for the stem versus the root as indexing terms.

Table 5.6: Mean Average Precisions for ZAD Collection

Index	MAP
No Stemming	0.4239
SAS Root	0.4764
SAS Stem	0.4948

It is clear that the stem outperforms the root in terms of precision and recall. This is a direct consequence of conflating all derived forms under one indexing term. To illustrate this fact more clearly, consider the root "slm", it would store the stem"muslim" with the pattern muCCiC and the stem "salAm" meaning

[3] http://lucene.apache.org.

"*peace*" with the pattern CaCAC under the same index even if the search was intended for the second. Therefore, retrieving documents containing the first word would reduce precision due to unrelated documents being added to the collection. The stem preserves the pattern, which is adequate to categorise a word according to its semantic class. However the number of indexing terms would grow to accommodate more derived forms, and this is a major disadvantage of stem indexing.

It is a possibility that mean average precisions are not strong indicators in revealing the significance of the tests, so statistical measures have to be used to make sure these numbers are not produced by chance [35]. The Wilcoxon Signed-Rank Test is used, and Table 5.19 shows the p-values [80]. The results are statistically significant and the stem is confirmed to be superior in performance to the root.

Table 5.7: Wilcoxon Signed-Rank Test

	$p - value$
Root vs. No Stemming	< 0.0001
Stem vs. Root	< 0.0001

It is fair to assume that web users do not go through all retrieved documents, and read only a subset of them. Table 5.8 shows the precisions at 5, 10, 15, and 20 cutoff points.

Table 5.8: Precisions for Top Ranked Retrieved Documents

Precision at	No Stemming	Root	Stem
5 Docs	0.3627	0.4373	0.472
10 Docs	0.344	0.4067	0.4427
15 Docs	0.3262	0.3911	0.4169
20 Docs	0.318	0.3707	0.4007

For the ZAD collection, Table 5.9 shows the precisions for the top 20 ranked retrieved documents

Table 5.9: Precisions for Top Ranked Retrieved Documents for ZAD Experiments

Precision at	No Stemming	Root	Stem
5 Docs	0.4917	0.5500	0.5250
10 Docs	0.3833	0.4167	0.4250
15 Docs	0.3167	0.3333	0.3722
20 Docs	0.2625	0.2833	0.3271

A comparison of the retrieval performance for the three indexing terms can also be done measuring precisions over 11 recall points. Figure 5.4 shows the curves for the three indices. It is clear that SAS stem method produces better precision numbers at all recall points.

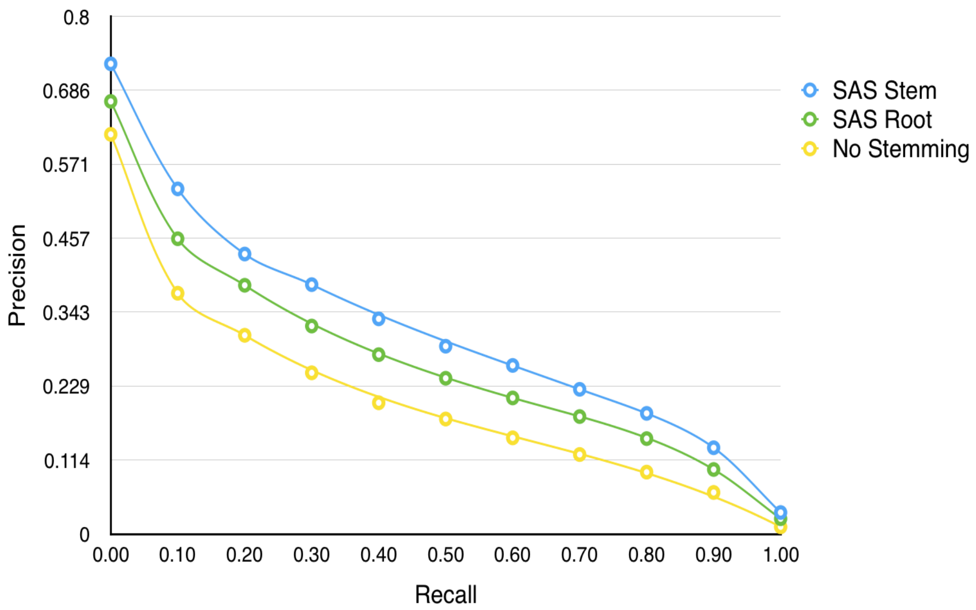


Figure 5.4: TREC Root vs. Stem Average Precision/Recall.

The SAS stem method outperforms the SAS root method as expected since semantically unrelated words are conflated under one indexing term. Furthermore, nature of Arabic linguistics and the root stemmers themselves add to the problem, as we will show in section 5.7.3.

The stem contributes to the elimination of any irrelevant documents, since it gives the retrieval system the ability to distinguish it from other derived forms via its morphological pattern. The word's root lacks this feature and using it as an indexing term can lead to retrieving irrelevant documents.

A similar conclusion was drawn based on the tests conducted on the ZAD collection. It is observed that the stem outperforms the root as an indexing term, and always produces better precision except for the first two recall points (0.00 and 0.10) as Figure 5.5.

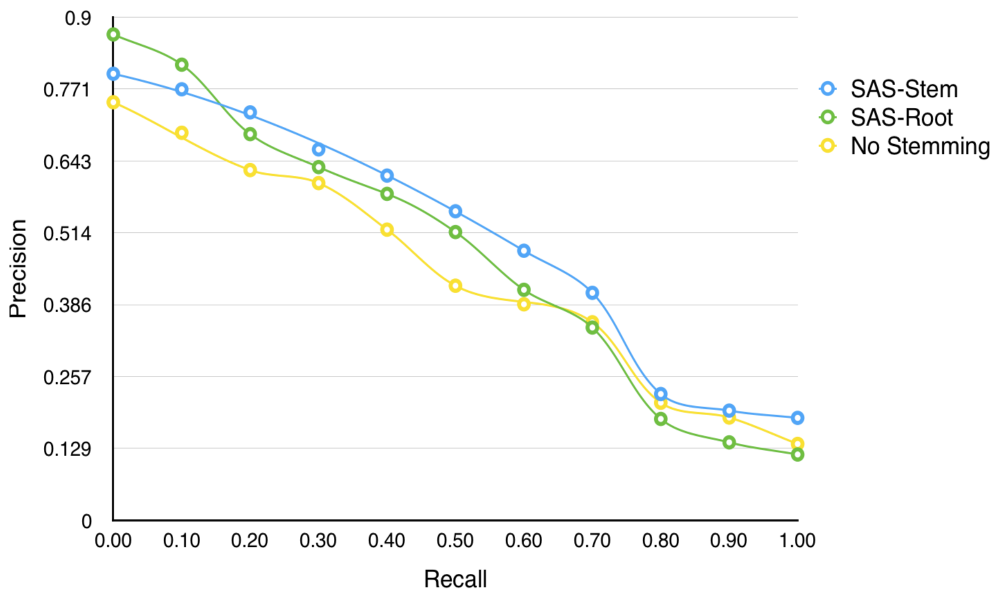


Figure 5.5: ZAD Collection Root vs. Stem Precision/Recall.

5.7.2 Khoja vs. Sebawai vs. SAS

An experiment to compare SAS to Khoja and Sebawai was conducted on the TREC collection (the results reported in this section have been published in CIKM'14 [5].) Table 5.10 shows the mean average precisions for the three stemmers compared to no stemming.

Table 5.10: Mean Average Precision

Index	MAP
No Stemming	0.1949
Sebawai	0.2206
Khoja	0.2245
SAS	0.2538

It is clear from Table 5.10, and according to Equation (5.3) that SAS root method recorded a 13% and a 15% relative gain over Khoja and Sebawai respectively.

$$Relative\,Gain = \frac{SAS\,MAP - Other\,MAP}{Other\,MAP} \tag{5.3}$$

For the ZAD Collection, the mean average precisions for the three stemmer are shown in Table 5.11.

Table 5.11: ZAD Collection Mean Average Precisions

Index	MAP
No Stemming	0.4239
Sebawai	0.3447
Khoja	0.4409
SAS	0.4764

Table 5.12 shows the precisions for the highest ranked documents. SAS gives better precisions at all cut-off points, whereas Khoja performs a little better than Sebawai.

Table 5.12: Precisions at Highest Ranked Documents

Precision at	No Stemming	Sebawai	Khoja	SAS
5 Docs	0.3627	0.4053	0.4133	0.4373
10 Docs	0.3441	0.3756	0.3867	0.4067
15 Docs	0.3262	0.3611	0.3716	0.3911
20 Docs	0.3180	0.3467	0.3540	0.3707

Table 5.13 shows the top 20 ranked documents for the three stemmers with regard to the ZAD collection. The Khoja stemmer outperforms the other stemmers according to the table

Table 5.13: Precisions at Highest Ranked Documents for ZAD

Precision at	No Stemming	Sebawai	Khoja	SAS
5 Docs	0.4917	0.4833	0.5500	0.5833
10 Docs	0.3833	0.3583	0.4333	0.4417
15 Docs	0.3167	0.2889	0.3639	0.3528
20 Docs	0.2625	0.2396	0.3146	0.2979

It was observed that all stemmers perform better with queries that have fewer stop words; the more stop words in a query, the worse the precision. This is normal since stop words are common amongst all documents, and they are initially removed from the list of words. So if a query had more stop words than derived words, the retrieval precision would be affected because fewer distinct terms were fed to the retrieval system .

All three stemmers produced low precision for query number 33, where the topic is related to "the impact U.S. military vessels in the Suez canal have on the region politically and economically". The reason is that there are three documents in the collection judged to be relevant and none of them is related to the subject. Two of them (19960918_AFP_ARB.0077 and 19971116_AFP_ARB.0145) bear reports about "Pittsburgh" submarine and the aircraft carrier "George Washington" crossing the Suez canal. The third document (20001101_AFP_ARB.0096) talks about the US military limiting its movement through the Suez canal as a result of the attack on USS Cole in October 2000. When the terms in the documents are stemmed to be indexed, they have no common roots with the terms in the query since the query is about the political and economical impact on the region, and the documents never mention these two aspects.

All stemmers also did poorly in query number 75, because the query contains two new foreign words, "kmbywtr" (كمبيوتر) meaning "*computer*", and "fyrwsAt" (فيروسات) meaning "*viruses*", and hence their absence from the lexicons justified why they had been stemmed incorrectly. The other words in the query have broad meanings and would not assist to narrow the search.

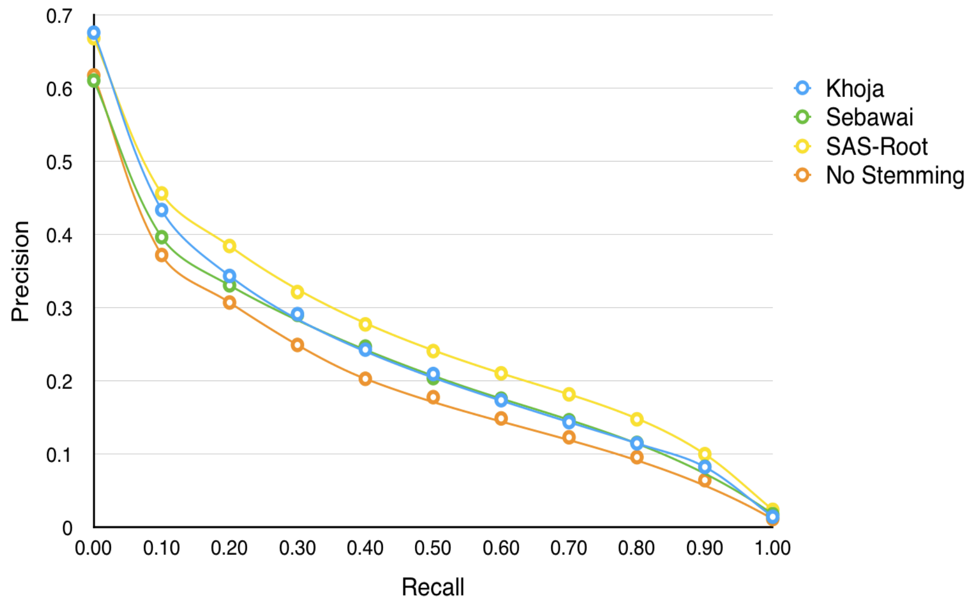


Figure 5.6: Khoja vs. Sebawai vs SAS Average Precision/Recall Graph.

SAS root method performs better than the other two stemmers, giving higher precision at all recall points except for the 0% recall. Also, above 20% recall, Sebawai is essentially the same as Khoja while SAS produces better precisions for the remaining recall points. SAS outperformed Khoja and Sebawai in both tests collections.

The SAS root method performed particularly well with short queries (fewer than 8 morphologically derived words). Strictly speaking, SAS produced high precisions with queries 61, 56, 48, and 59 containing 3, 4, 7, and 7 derived words respectively.

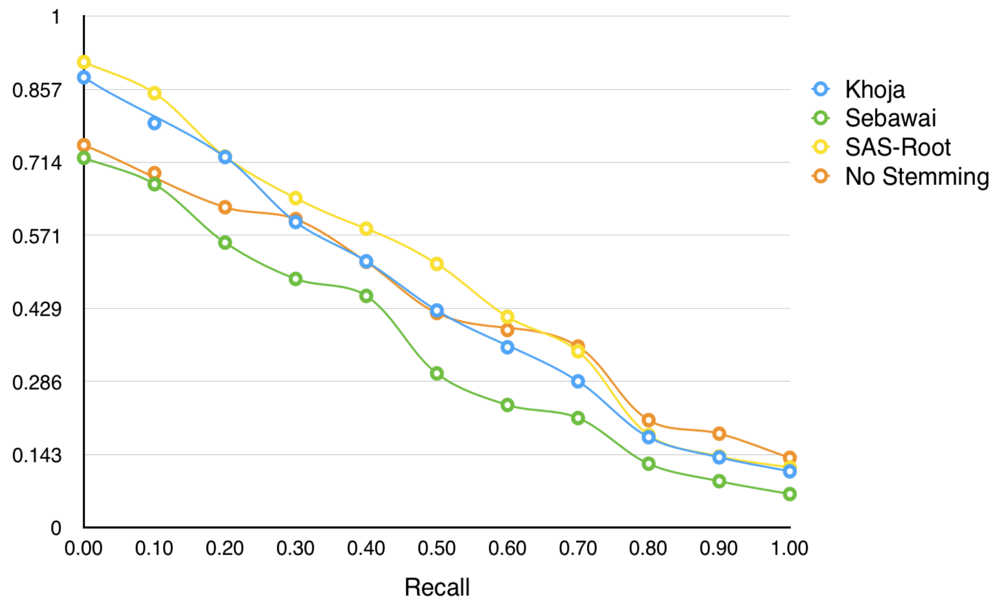


Figure 5.7: Khoja vs. Sebawai vs SAS Average Precision/Recall for ZAD.

For example, Khoja and Sebawai stemmed all derived words in query 48 correctly except for "trY" (ترى) meaning "*she sees*". Khoja returned the word intact (not stemmed), whereas Sebawai produced "rwy" (روي). SAS produced the root “r>y” (رأي), which was the correct solution. Figure 5.7 shows the same 11-point average precisions graph for the ZAD collection.

It is worth noticing that the no-stemming (unprocessed words) overlaps and sometime outperforms the three root indices. This might be due to the fact that the root conflates many stems under one index, most of these stems are not semantically related. For instance, the words “sullam” (سُلَّم) “*ladder*”, and “mustaslim” (مُستَسلِم) “*he who gives up*”, share no semantic features with each other, and yet both words are indexed under the same term, namely the common root “slm” (سلم).

5.7.3 Issues in Arabic Root Stemming

In order to analyse the points of success and failures for each stemmer, we randomly selected a sample of 100 words to measure the accuracy of the stemmers. Each word was stemmed by the three stemmers, and then checked manually. Table 5.14 shows the percentage of correct roots for each stemmer. The errors fall into one of the following categories:

- Over-Stemming (OS): This occurs when a root’s radical is deemed an affix and is stripped off as a result.
- Under-Stemming (US): When an extra letter that is supposed to be stemmed is left as a root’s radical.
- Out-of-Lexicon (OOL): When a word contains a morphological unit that is not in the lexicon, so no stemming occurs.

Table 5.14 shows that Khoja and Sebawai stemmers generates more over-stemming errors than SAS. This is expected since prefixes and suffixes are stripped off blindly in these methods.

Table 5.14: Correct Solution and Error Types

Stemmer	Correct	OS	US	OOL
Khoja	80%	14%	5%	1%
Sebawai	79%	15%	2%	4%
SAS	84%	4%	10%	2%

SAS checks first for the legality of the prefixes and suffixes for a particular pattern before stemming them, which is why it has few over-stemming errors (4%). However, it has the most under-stemming errors. SAS obviously needs a lexicon expansion to include more legal prefix-pattern-suffix combinations in order to overcome the under-stemming problem. Out-of-lexicon errors for SAS indicate that the lexicon does not contain the roots because there was no case found where the pattern was not recognised.

There are common errors for the three stemmers when it comes to over-stemming, but SAS has the fewest errors in this category. Some of these errors involve the lexical components of the stemmer, whereas other errors are just unnecessary due to removing affixes blindly. In general, over-stemming errors occur under three conditions:

1. when a root starts or ends with a letter that can be deemed an affix,
2. when one word matches two patterns, and
3. when multiple prefixes are used in one word.

The first condition happens with roots whose first or last radical could also match a popular prefix or suffix respectively. All stemmers faced this dilemma, but SAS had been the least affected as a consequence of a validation step taken prior to stripping off affixes, where SAS checks the compatibility table for which prefix goes with what suffix.

Consider the word “wjhhn” (وجههن) meaning “*their faces*”, where “*their*” here refers to a feminine pronoun. The word is supposed to be segmented as “wjh” (“*face*”) and “hn” (“*their*”), however Khoja judged the first consonant “w” (“*and*”) to be a conjunction particle and stemmed it hence. This is a reasonable judgement given this prefix frequency in MSA, but the stemmer was left with two letters, it rectified the root by appending the vowel “y” to become “jhy” instead of the correct solution “wjh”.

SAS outperformed Khoja and Sebawai whenever an illegal prefix-suffix combination was used. SAS compatibility table is effective because it restricts prefix-suffix combinations to those allowed in the Arabic language.

Consider the word “bAlqAnwn” (بالقانون) meaning “*by the law*”. It should be segmented as the preposition particle “b” meaning “*by*”, the definiteness marker “Al” “*the*”, and the stem “qAnwn” “*law*”. This stem matches the pattern CACwC and produces the root “qnn”. Due to the fact that the word ends with “wn”, which is a very popular masculine plural suffix, Sebawai and Khoja produced the root “qwn”. The reason that SAS did not make the same error is because of the compatibility table, since the preposition particle “b” mandates the existence of a suffix reflecting a genitive case, namely the suffix “yn”, so the combination “bAl”-“wn” is deemed invalid and the suffix “wn” remains intact as part of the stem.

The second over-stemming condition occurs when a word matches two patterns, and thus produces two correct roots. This is expected in MSA, and it is safe to assume all three stemmers, including SAS, suffered from this weakness. For example, consider the word “mwA$y” (مواشي) meaning the plural of “*livestock*”. This matches the pattern mCACC, producing the root “w$y” (وشي), which was SAS and Khoja’s choice, whereas “wA$” (invalid root) was Sebawai’s. The same word “mwA$y” could also match the broken plural pattern CwACC with the root “m$y” (مشي), and this is the correct solution. The stemmer had difficulty selecting the right answer, and for Khoja, it would always be the first solution because the pattern mCACC comes before the pattern CwACC according to Khoja’s list of patterns.

For SAS, the word “tjAry” (تجاري) would match the verbal pattern tCACC with the root “jry” (جري) to mean (“*it/she competes*”), however the correct segmentation is composed of the nominal pattern CCAC with the root “tjr” (تجر) and the suffix “y”, to mean (“*commercial*”). SAS produced the wrong solution because of a process that gave precedence to weak roots when a selection had to be made amongst multiple analyses.

There are certain affixes that, when combined with stems for inflection, match other words. For instance, the prefix “>” as in “>nAm” (أنام) would match the verbal pattern >CCaC and the root “nwm” (نوم) to mean “*I sleep*”. It would also match the nominal pattern CaCAC with the root “>nm” (أنم) to mean “*all creatures*”. It is worth pointing out that such a phenomenon is observed exclusively with weak roots.

Using multiple prefixes or suffixes in Arabic is very common, and it is sometimes necessary to use more than one prefix to reflect the future tense in verbs, or more than one suffix to represent pronouns. This is the third condition in which over-stemming occurred.

For example, consider the word “fsymknhm” (فسيمكنهم), meaning “*then he/it will enable them*”. The word has more than one segmentation, but the correct one is “f”+“s”+“y”+“mkn”+“hm”. In other words, the combination should consist of the conjunction particle “f” (“*then*”), the future verbal marker “s” (“*will*”), the prefix “y” indicating a 3rd person.masculine.singular subject (he/it) for the verb "mkn" "*enable*", and the pronoun "hm" marking a 3rdperson.masculine.plural object ("*them*"). Instead the Khoja stemmer produced the root "swm" after considering the verbal marker "y" as one of the radicals using the combination "f"+"sym"+"knhm". This combination includes the conjunction particle "f", the verb "sym", the subject pronoun 3rdperson.feminine.plural "kn", and the object pronoun 3rdperson.masculine.plural "hm" ("*them*"), but it is not the correct morphological combination. Khoja altered the second radical of the produced root "sym", since no such root exists in Arabic, to "w", and so the root becomes "swm" with two missing radicals.

Under-stemming occurs when a letter that is supposed to be stemmed is left as a root's radical, and all stemmers had been affected by this ambiguity. There are certain linguistic conditions that sometime create this situation; in particular the assimilation of one vowel in a stem can mislead the stemmer into considering an affix as a root's radical.

The word "SmthA" (صمتها) meaning "*I fast them*", or stopped eating and drinking during them, with a combination of two suffixes "t" indicating 1stperson singular subject and the 3rd person feminine singular object pronoun "hA", which in Arabic can be used for humans and things–in the context here, it refers to the days. Since the second radical of the root "Swm" (صوم) is the vowel "w", which is assimilated and disappears from the surface as a result of a certain phonological rule, the suffix "t" replaces it as a radical to generate another common root "Smt" (صمت) "*related to silence*", and only the suffix "hA" is removed. It is worth noting that such cases can only be observed with weak radicals.

Under-stemming could also occur as a result of over-stemming, in other words, removing one radical would encourage the stemmer to replace that missing radical with an affix. An example of this case is when a root has radicals that can be part of a pronoun as in "lyHtkmA" (ليحتكما) "*they would set in court*", "*they*" referring here to 3rdperson.masculine.dual subject. There is a suffix "tkmA" (تكما) in Arabic, therefore it was removed from the word to leave only one radical "H" of the correct root. The stemmer produces the root "lwH" (لوح) instead of

“Hkm” (حكم) because the two radicals, “k” and “m”, have been compensated by “l” and “w”.

Out-of-lexicon errors for Khoja pointed to missing roots and patterns. There are very popular patterns in MSA that are missing from the Khoja lexicon; the patterns CACwC and tCACyC are two examples. The first is the noun of instrument pattern that is commonly used in everyday language, and usually refers a tool or a device, whereas the second is a very common broken plural. Also the root “gTy” is missing from the lexicon, and this is the root of a popular word “>gTyp” (أغطية) meaning “*covers*”.

Sebawai also needs the addition of more prefix-suffix combinations to its lexicon. For instance the word “>fbnEmp” (أفبنعمة) would generate “fbnEm” (فبنعم), which is an invalid root that is not part of the Arabic root lexicon, nor does it exist in Sebawai’s lexicon. The missing combination, which is composed of an interrogative particle “>” meaning (“*is it?*”), a conjunction particle “f” with a meaning close to (“*then*”), and a preposition particle “b” meaning (“*by*”), is the reason that Sebawai couldn’t reach the correct analysis. This combination was not included in the data that Sebawai was trained on, and therefore it was not added to its lexicon

5.7.4 Suggested Solutions

Based on Arabic stemming challenges explained in section 5.7.3, solutions can be classified into three types; hand-encoded, lexicon-related, and external. There are words in the Arabic language that need to be hand-encoded, simply because there is no rule that can be applied to them, so they constitute rules by themselves [54]. The lexicon-related processing involves expanding the lexicon to include more supporting functionality and components such as lists of the most common patterns and affixes in Arabic.

The external processing pertains to the fact that MSA lacks short vowels, which compounds the ambiguity problem, and calls for leveraging a syntactical and semantical analyser to disambiguate problematic words. It can be argued that for IR, such a solution might not be necessary since a query may consist of keywords written in a sequence, and they might not necessarily form a sentence by themselves.

Weak roots are the source of ambiguity for most of the cases presented in section 5.7.3, it is worth mentioning that we only address roots for which two of the three radicals are vowels. Those roots have special place within the Arabic phonology mainly because radicals are assimilated, take new shapes, or disappear totally from the surface form. These double-weak roots, as they will be referred to hereafter, are rare according to our lexicon; there are fifty four roots in the nominal branch out of 1413 roots.

It is a fact that double-weak roots are similar to each other with relation to how they behave when undergoing certain phonological operations. This is trivial from a computing perspective because instead of hand-encoding every root, a rule can be applied on the paradigm for deriving new words with the inflectional operations performed later.

Consider the word “yarah” (يَرَهُ) with the Quran Index (QI) (99.7.6) and the root “r>y” (رأي), meaning “*he sees it/him*”. This is correctly segmented as ya+ra+h, where the first segment “ya” is the imperfective prefix indicating a masculine subject, and the last segment is the pronoun “h” representing a singular masculine object. This leaves only the consonant “r” as the verb in this case and the last two vowels have been assimilated and omitted on the surface. The two missing radicals have to be restored to form a correct root.

In MSA the same verb would be written as “yrAh” (يراه), where the long vowel “A” is added to indicate the presence of a vowel as one of the root’s radicals, in this case ">" (أ). The third radical "y" is missing and has to be restored.

Consider another double-weak root, "wqy" (وقي), where the two vowels are separated by a "q", with the nominal pattern CaCCaY (فعلى) would produce the stem "taqwaY" (تَقوَى) QI (49.3.6), which contains an extra consonant "t", which is not part of either the pattern or the root. It is a rule that the "t" can replace the vowel "w" in certain patterns.

Since the number of double-weak roots in the Quran is less than a hundred, it is highly economical to hand-encode all of these roots in the lexicon. What makes the Xerox Analyser stand out is the fact that all four thousand nine hundred roots had been hand-encoded with all legal patterns. The SAS set has smaller subset and it should not consume many resources in terms of both preparation and development.

Lexicon-related solutions would ideally include lists of the most frequent patterns in order to select a single analysis when a multi-analysis case is encountered. However, this solution requires the existence of a large corpus in order to learn the most common combinations (patterns and roots) in the Arabic language. As we saw in section 5.7.3, the word "tjAry" had two analyses with two different roots. In this case it is more accurate to include a list that should point out that the pattern CCAC and the root "tjr" is the right analysis because it occurs more frequently.

5.8 The Evaluation of the Stem as an Indexing Term

The stem in Arabic Morphology represents a unique identifier because no two words could have the same stem (pattern and root) and differ in meanings. Unlike the root, where many unrelated words can be grouped under one term, the pattern within the stem structure distinguishes it from other derived words and gives it its own semantic meaning within the mental lexicon. For instance, the stem "dAris"

(دَارِس) with the pattern CACiC and the root "drs" could only mean a "*student*", whereas the stem "madrasap" (مَدرَسَة) with the pattern maCCaCap (noun of place) can only point to a place where lessons are being taught, or a school. That is the reason why the stem is preferable for IR applications, or for any NLP application for that matter.

Table 5.15: Light Stem Mean Average Precisions

Index	MAP
SVM Lexem	0.2933
MADA	0.2911
SAS	0.3035
light10	0.3048
Al-Stem	0.2584
No Stemming	0.1949

It has been shown that the stem is more efficient in terms of performance than the root [8; 53; 68]. Experiments were conducted using our SAS light stemmer, light10, and Al-Stem, SVM and MADA stems. Table 5.15 shows the mean average precisions for the five stemmers. It comes as no surprise that light10 slightly outperforms SAS knowing that light10 was developed based on the TREC corpus, which we are using for evaluation. The most common prefixes and suffixes in the TREC corpus have been used as the lexicon for light10.

Although Al-Stem was also trained on the same corpus, its patterns are not according to Arabic morphology. That is the reason why Al-Stem's performance is inferior to the other two stemmers. Al-Stem might remove only suffixes and leave prefixes, as in "wllmslmyn", "wbAlmslmyn", and "wkAlmslmyn" where they supposed to be stemmed as "mslm"; however after removing the suffix, Al-Stem produces three different stems, "wllmslm", "wbAlmslm", and "wkAlmslm" respectively. Furthermore, some common suffixal combinations are missing: "Athm" is a combination of two suffixes, "At" signifying a feminine plural suffix, and "hm" representing the possessive pronoun for 3rd person masculine plural.

The original corpus on which Al-Stem was trained did not include such combination, and that in turn impacted Al-Stem's performance negatively.

For the ZAD collection, Table 5.16 shows the mean average precisions for all stemmers run on ZAD collection.

Table 5.16: Light Stem Mean Average Precision for ZAD Collection

Index	MAP
SVM	0.5433
MADA	0.5404
SAS	0.4948
light10	0.4926
Al-Stem	0.4758
No Stemming	0.4239

It is fair to assume that web users do not go through all retrieved documents, but read only a subset of them. Table 5.17 shows the precisions at the 5, 10, 15, and 20 document cutoff points.

Table 5.17: Precisions for Top 20 Retrieved Documents for the TREC Collection

Precision at	No Stemming	SAS	light10	Al-Stem	MADA	SVM
5 Docs	0.3627	0.4720	0.4960	0.4427	0.4853	0.4693
10 Docs	0.3440	0.4427	0.4493	0.4080	0.4293	0.4347
15 Docs	0.3262	0.4169	0.4382	0.3884	0.4080	0.4036
20 Docs	0.3180 0	0.4007	0.4147	0.3727	0.3980	0.3913

Table 5.18 shows the top 20 ranked documents for the light stemmers run on the ZAD collection. It is interesting to observe that SAS outperforms light10 at 15 and 20-document cut-off, while at 5 and 10-document cut-off, light10 surpasses SAS slightly.

Table 5.18: Precisions for Top 20 Retrieved Documents for the ZAD Collection

Precision at	No Stemming	SAS	light10	Al-Stem	MADA	SVM
5 Docs	0.4917	0.5250	0.5500	0.5333	0.6667	0.6750
10 Docs	0.3833	0.4292	0.4292	0.4250	0.5250	0.5333
15 Docs	0.3167	0.3750	0.3500	0.3639	0.4306	0.4278
20 Docs	0.2625	0.3292	0.3042	0.3188	0.3625	0.3604

It is a possibility that mean average precisions oversimplify the tests, so statistical measures have to be used to make sure these numbers are not produced by chance [35]. The Wilcoxon Signed-Rank Test is used, and Table 5.19 shows the p-value for Al-Stem vs. SAS stem method [80]. The results are statistically significant and SAS stem method is confirmed to be superior in performance to Al-Stem.

Table 5.19: Wilcoxon Signed-Rank Test

	$p - value$
SAS vs. Al-Stem	< 0.001

Figure 5.8 shows the precision over 11 recall points graph for the three stems. SAS and light10 almost overlap each other, whereas Al-Stem does not compete with them at all. Light10 does not have any verbal prefixes within its lexicon, and hence it leaves verbs intact unless it has a common suffix with nouns (e.g., "wn").

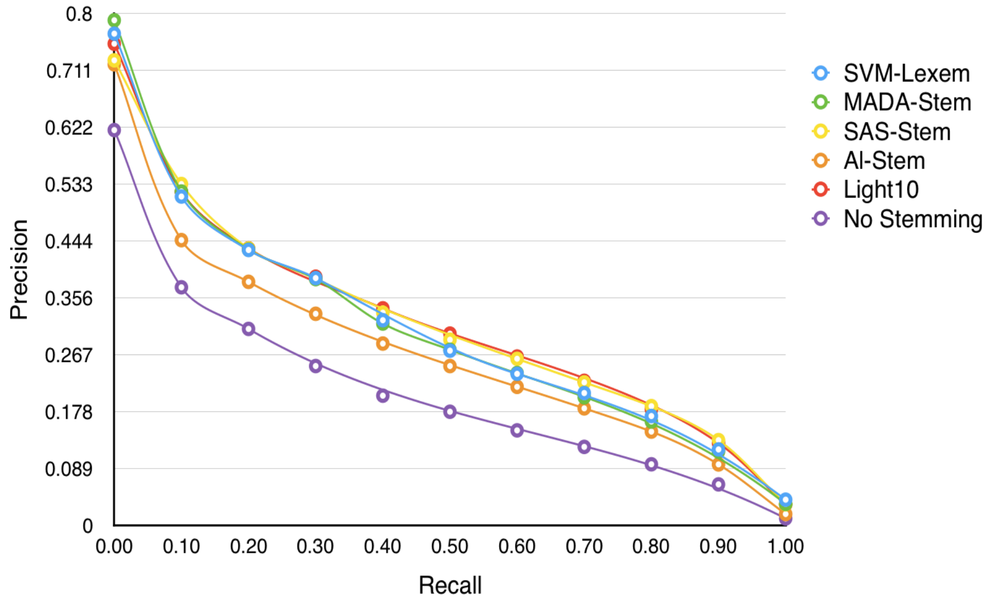


Figure 5.8: Average Precision and Recall for the Stem Experiment.

In spite of the finding that light10 performs slightly better than SAS stem method, this is not conclusive because the evaluation corpus was the same on which light10 was trained. Extra testing on the ZAD collection was conducted, and the findings support our claim, that is the SAS method outperformed light10 producing a slightly better mean average precision.

Figure 5.9 shows the precisions for the five light stemmers over 11 recall points.

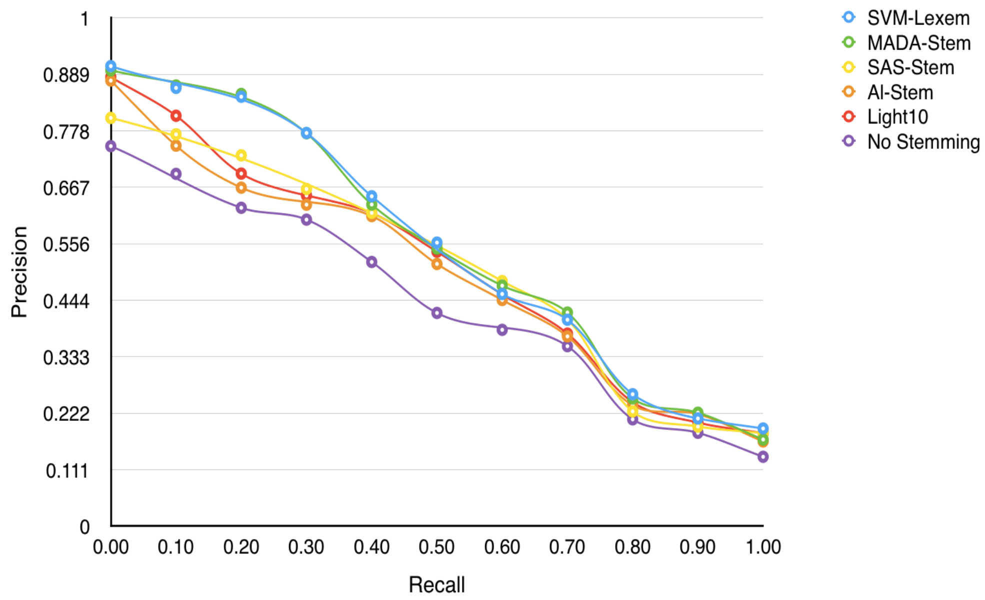


Figure 5.9: Average Precision/Recall for the Stem Experiments on ZAD.

5.8.1 Issues in Arabic Light Stemming

We have selected 300 random words from the ZAD corpus to measure the performance of the five light stemmers. The results revealed the weaknesses of each stemmer, and gave a reasonable view of the kinds of errors produced by each stemmer.

Table 5.20: The Five Stemmers' Performance Using 300 Random ZAD Words

Stemmer	Correct Analysis	%
SVM	207	69%
MADA	230	77%
SAS	223	74%
light10	120	40%
Al-Stem	141	47%

As Table 5.20 shows that MADA performed the best followed by SAS, the other three stemmers recorded a successful rate of $< 70\%$, the discussion below illustrates the weak points of each stemmer and suggests ways to avoid some of the made mistakes. Moreover, the results show that a stemmer should have a good lexicon, which should include either stems (e.g., BAMA) or patterns (e.g., SAS) for it to cover the language vocabulary well.

The types of errors MADA has committed are shown in Table 5.21. The first is pertaining to removing the vowel “A” from some nominal stems even if it is part of the morphological composition of the word. The word in the example shown has the pattern “AstCCAC” (استفعال), which is the infinitive noun (المصدر) of the tenth verbal form “AstCCC” (استفعل). As a consequence, the vowel “A” is part of the stem — just as the two other consonants; namely “st” (ست), and should not be deemed a prefix.

Table 5.21: Types of Errors for MADA Stemming

Error	Example
Reckoning the vowel “A” as a prefix	“wAstgfArhm” → “stgfAr” واستغفارهم ← ستغفار
Vowel stem	“lqst” → “qss” لقست ← قسّ
Out of BAMA lexicon	“AlArtDAE” → “AlArtDAE” الارتضاع ← الارتضاع
Replacing the vowel “A” with “>” in verbs	“ADrb” → “>Drb” اضرب ← أضرب

The second error occurs with all stemmers with no exception, vowel stems — stems that have a vowel as one of their root’s radicals. In this example, the root is “qsw” (قسو), but the vowel at the end is assimilated due to a phonological rule, and thus the first and the second radicals are merely present on the surface. This issue poses the hardest challenge faced by researchers in terms of Arabic stemming, some suggested hard-coding these forms in the lexicon in order to be highly confident of the stemmer’s coverage [54].

The AMIRA method shares the types of errors that MADA has generated, however AMIRA has also produced it own errors. As Table 5.22 shows, the first type happens when AMIRA replaces the vowel “A” with the vowel “>”, which alters the word’s meaning. The example demonstrates that AMIRA deemed the consonant “k” as a preposition particle, which led to substituting the vowel “>” in place of the vowel “A”, and therefore producing a new stem that has no common morphological units with the original word.

Table 5.22: Types of Errors for AMIRA Stemming

Error	Example
Treating the vowel “A” as “>”	“kAsrp” → “>srp” كاسرة ← أسرة
Changing the verbal tense from present to past	“ybyEh” → “bAE” يبيعه ← باع
Replacing a plural with its singular	“AlqwAEd” → “qAEdp” القواعد ← قاعدة

The second error changes the tense of the verb from present to past, so instead of generating the stem “ybyE” (يبيع) “*he sells to him*”, the stem “bAE” (باع) “*he sold to him*” is produced. The third type of errors happens because AMIRA substitutes any broken plural with its singular form after stripping prefixes and suffixes off.

The word “AlqwAEd” translated to the English language as “*the rules*” is changed to “qAEdp” “*a rule*”. This might be helpful in certain NLP applications, but it does not certainly suit an Information Retrieval application.

Table 5.23 shows the errors that SAS method has generated, the first type occurs because SAS lexicon does not include certain prefix-pattern-suffix combinations.

Table 5.23: Types of Errors for SAS Stemming

Error	Example
Out of lexicon	“bmErfthmA” → “bmErfthmA”
	بمعرفتهما ← بمعرفتهما
More than one analysis	“EmlAn” → “EmlAn”
	عملان ← عملان
Removing the suffix “p”	“AstbAnp” → “AstbAn”
	استبانة ← استبان

The prefix “b” and the suffix “thmA” are not part of the legal combinations designated for the pattern “mCCC”. The second type of errors arises when more than one morphological analysis is produced. The third error in Table 5.23 alters the meaning of the word; it changes the word’s part of the speech from a noun to a verb, and that could impact the obtained search results negatively.

The Al-Stem had made similar errors as the previous three stemmers, but it had also committed unnecessary errors, making no distinction between verbal and nominal prefixes is an example illustrated in the first row of Table 5.24. The example shows that the verbal prefix “t” is removed from the noun, and as result of that, a new stem is produced.

Table 5.24: Types of Errors for AlStem Stemming

Error	Example
Treating the letter "t" as a prefix in nouns	"btxSySh" → "xSyS"
	بتخصيصه ← خصيص
Missing Suffix	"EdAwthmA" → "EdAwthm"
	عداوتهما ← عداوتهم
Multi-prefix	"fbHkmth" → "fbHkm"
	فبحكمته ← فبحكم
Blindly stripping suffixes off	"wAljnwn" → "jn"
	والجنون ← جن

Missing affixes from the lexicon caused Al-Stem to produce incorrect stems posing a major issue with regard to Al-Stem's performance. This can be rectified by adding the missing affixes to the lexicon. Furthermore, Al-Stem faced a problem when dealing with multi-prefix words, the third row in Table 5.24 demonstrates this clearly. The example shows that the word "fbHkmth" (فبحكمته) "*then by his wisdom*" has two prefixes, namely the conjunction particle "f" "*then*" and the preposition particle "b" "*by*", but Al-Stem removes the first and leaves the second as part of the stem. This type of errors can be dealt with by running through the list of prefixes sequentially in a loop after ordering the list according to Arabic Morphology.

Besides sharing some errors with the other stemmers, the light10 introduced new error types. Table 5.25 shows that going through a list of suffixes in the wrong order could have negative consequences on the stemming results. The light10 stemmer has the suffixes list ordered incorrectly; it gives case-marking suffixes ("An, wn, yn") precedence over possessive pronouns ("h, y"). It is a rule in Arabic morphology that possessive pronouns are placed in the last position of the word.

Table 5.25: Types of Errors for Light10 Stemming

Error	Example
Wrong firing order of suffix removal	"mqdwrAth" → "mqdwrAt" مقدوراته ← مقدورات
Missing nominal affixes	"brfq" → "brfq" برفق ← برفق
Missing verbal affixes	"ftrAdft" → "ftrAdft" فترادفت ← فترادفت
Blindly stripping suffixes off	"fAlmEyn" → "mE" فالمعين ← مع

There exists a suffix that includes both the case-marking suffix "At" and the possessive pronoun "h" to become "Ath", but there could never be a suffix of the reverse order; strictly speaking, the suffix "hAt" simple does not exist in the Arabic lexicon. Row 1 in Table 5.25 illustrates how this error could have been avoided. In order to remove the suffix "Ath" from the word "mqdwrAth" (مقدوراته) "*his assets*", the 3rd person masculine singular possessive pronoun "h" "*his*" should have been placed in the list of suffixes before the feminine plural suffix "At".

The last error type is shared with Al-Stem: the light10's lack of knowing the word's pattern impacted its capacity to recognize correct stems. The stemmer succeeded in removing the prefix "fAl", but failed in reckoning the last two letters of the word "yn" as a suffix because these two letters are part of the stem "mEyn" فمعين ("*who lends a hand*"). Instead, the light10 stemmer produced the stem "mE", which is easily confused with the preposition particle "mE" (مع) "*with*". This error can be avoided if the stemmer includes patterns in its lexicon, so the word can be easily matched to the pattern "mCCC", and the suffix "yn" would be left intact.

5.9 Conclusion

We presented the numerical findings of the different stemming methods being evaluated in this thesis in an attempt to gauge their performance. We ran the stemmers on two collections; the TREC collection to measure how effective the stemmers do with the Modern Standard Arabic (MSA), and the ZAD collection to gauge the stemmers performance with the Classical Arabic Language. Our root finding method outperformed Khoja and Sebawai in both collections. We conjecture that SAS stem method would surpass light10 in terms of precision if they were to be tested on a new corpus. And as anticipated, the SAS method outran the light10 when tested on the ZAD corpus, and it was evident that the light10 needs lexicon expansion to include patterns in order for it to compete well with the SAS method.

Chapter 6

Conclusion and Future Directions

This thesis explored the impact of Arabic morphology on the performance of Arabic Information Retrieval (AIR) by assessing which indexing method (root vs. stem) was more effective in terms of precision. Moreover, the thesis proposed a new root-finding algorithm tailored towards enhancing the performance of the retrieval system while utilising fewer lexical items. The findings of the thesis provide evidence to support the claim that a robust end-to-end root solution can be developed with a smaller lexicon than what was previously presented in the field.

The concept of stemming Arabic in accordance with a predetermined set of constraints enforced on the tokenisation process has also been investigated. The thesis asserted the assumption that in Arabic, lexical components have explicit relationships dictating which elements are permitted to be combined to form surface words. For instance, not all prefixes can be arbitrarily grouped with any pattern or suffix to form a word, and hence a legal prefix-pattern-suffix combination is the only assurance that a correct root can be found.

With highly inflective languages such as Arabic, where many morphemes are allowed to be combined to form a single word, the segmenting rules proved to be most effective especially with the over-stemming problem, which occurs when a letter that is supposed to be part of the root is deemed an affix and stripped off as a result. Our proposed root-finding algorithm delivered a 13% Mean Average Precision (MAP) relative gain over state-of-the-art stemmers, and has produced the fewest number of over-stemming errors with 4% in comparison to 14% and 15% for Khoja and Sebawai respectively.

The work presented in this thesis tackles the fundamental structure of Arabic morphology, namely the root, from which all forms are generated. The root is the only determining factor to distinguish Arabic from foreign words, and as a consequence, it ought to be extracted in order to learn other features such as Part-Of-Speech POS tag or a nominal/verbal pattern. Therefore, our solution could be of great assistance to other NLP applications, text categorisation and data mining as examples. Furthermore, our solution can easily be embedded within such applications.

6.1 Contributions

The major contributions of this thesis to the Arabic Computational Linguistics field are:

- The evaluation of two indexing methods (root vs. stem), and presenting the empirical findings to prove the stem superiority over the root in terms of precision. The LDC Arabic corpus, which contains 383,872 tagged documents, 75 queries, and 10,031 manually-judged documents, along with Lucene search engine, have been employed in the experimental testing.

- A novel root-stemming method that leverages Arabic morphological rules in segmenting the word into valid morphological units. The algorithm introduced a set of constraints that determine the morphological borders amongst morphemes within a surface word, and only a valid prefix-stem-suffix combination is considered for the root extraction procedure. The segmentation rules set had been built in accordance with the Quran morphological guidelines, whereby the decomposition of the Quran words led to deriving a concise lexicon capable of producing the root better than Khoja and Sebawai.

- The comparison of our root-finding method against two leading root stemmers; Khoja and Sebawai. Our method proved its ability to find the root of an Arabic word, delivering 0.2538 mean average precision as shown in Table 6.1, while utilising a much smaller root lexicon.

Table 6.1: Mean Average Precision

Index	MAP
Sebawai	0.2206
Khoja	0.2245
SAS	0.2538

- A simple and yet more effective approach to designing a relatively compact lexicon to support Arabic Natural Language Processing, when previous solutions utilised huge lexicons (i.e., more than 4000 roots). Through the work presented in this thesis, it was feasible to find the roots for most of the Arabic vocabulary while employing a compact lexicon as Table 6.2 shows.

Table 6.2: Lexicon Numerical Comparison

Stemmer	Root	Pattern	Prefix	Suffix
Khoja	4,748	46	16	28
Sebawai	10,405	245	215	290
SAS	2,520	110	84	183

- A detailed description for an end-to-end root-finding solution, which employed a prefix-pattern-suffix compatibility table, using a high-level programming language (Java), and how to integrate that within an IR application using Lucene search engine from Apache, as it was detailed in Section 4.5.

Although the SAS method has been leveraged within an IR application, the Java-based stemmer can be used within other applications. It can be very helpful in text categorisation, where several words, which share the same root or the same pattern, are grouped under one common morphological unit.

6.2 Limitations

Arabic morphological analysis has challenges that fall into two categories; orthographic and phono-morphological. The first involves the shapes of the alphabets, and how different letters can often be written in place of others. Modern Standard Arabic (MSA) adds another difficulty to the second category due to the fact that short vowels are omitted, which compounds the ambiguity problem. In this section, both types of limitations are presented and in section 6.3, we speculate on how some of them may be overcome.

- SAS is limited only to Modern Standard Arabic (MSA) and Classical Arabic (fully vocalised) words are processed after the removal of short vowels.
- SAS performs as expected with sound roots (no long vowels), although it faces a situation where more than one analysis can be generated with weak roots. A selection of the correct root has to be made and currently the selection process merely considers the popularity of the pattern as a determining factor.
- The legal morphological combinations (prefix-pattern-suffix) table is not comprehensive; there are currently 5,617 entries in this table and more legal combinations need to be added to it. The root would not be recognised unless a legal combination for that particular word exists in the compatibility table, so the missing combinations hinder the performance of the solution.
- SAS has been tested using the TREC-11 Arabic corpus, which was created from one source (AFP). It is a known fact that most of AFP writers descend from North African countries (Algeria, Tunisia, and Morocco) and somewhat share the same writing styles.

- The TREC-11 Arabic corpus that was used in evaluating our solution contains many typos, and certain letters are replaced with others whose shapes are similar. For instance, the letter "h" ("ـه") would often be used in place of "p" ("ـة"), likewise the vowel "Y" ("ى") is used in place of the vowel "y" ("ي"). This had resulted in producing incorrect roots using our solution.

- There are certain letters that cannot follow each other due to the difficulty of pronouncing them in that sequence. For instance the letter "t" (ت) can not follow the letter "D" (ض). So if there is a pattern that contains the "t" (e.g., "muCtaCaC") and a root that starts with the letter "D" (i.e., "Drr") ("مُفتَعَل" and "ضرّ"), then the "t" ("ت") is replaced by a "T" ("ط") to become "muDTarr" ("مُضطَرّ") instead of "muDtarr" ("مُضتَرّ"). SAS currently is not capable of recognising such patterns, and they must be added to the lexicon.

6.3 Future Directions

The work presented in this thesis represents the foundation on which more elaborate effort can be undertaken. The thesis establishes the grounds on which we can develop more morphological components such as embedding short vowels to deal with Classic Arabic, and inventing new methods to assist in resolving the ambiguities posed by weak roots and patterns that contain long vowels. Although the methods employed in the root-finding solution are simple and straightforward, they represent the minimal operations pertaining to measuring the difference between the root and the stem as indexing terms. There is a need for more functionality that deal with the other morphological aspects; the relationship between nominal patterns is one example. Moreover, the lexicon we utilised is concise and needs more items to be added to it—more legal prefix-pattern-suffix combinations to be precise. Here are the major items that will be worked upon in the near future:

- Expanding the lexicon to include more legal prefix-pattern-suffix combinations via stemming new words and learning their morphological structure.
- Devising a mechanism to inter-digitise allowed roots with patterns; it is a well-known characteristic of the Arabic lexicon that not all roots can be used with every pattern. This major enhancement should improve performance immensely since it would automatically eliminate incorrect combinations, and hence minimise the possibility of incorrect roots being produced.
- Developing a more robust selection process for weak (voweled) roots to learn what vowel, if any, is assimilated when certain patterns are employed in the generation process. THIS IS JUST FOR TESTING This will require hard-coding all weak roots that can be used with patterns, which contain other vowels as part of their structure (e.g., "C**A**CiC").
- Adding the short vowels (a, u, i) to the patterns lexicon in order to accommodate Classical Arabic vocabulary. Embedding the short vowels within our lexicon is no trivial task, and it is a long-term project. This task would involve taking into consideration the impact these short vowels have on weak roots, and adjust the surface form accordingly. It might assimilate some vowels, or change their shapes to other vowels (e.g., "**w**qt" and "mi**y**qAt").
- Evaluating SAS using more textual resources to cover a wider range of Arabic written styles.

6.4 Concluding Remarks

Arabic stemming is no trivial task due to the huge number of derivational and inflectional operations that exist in Arabic morphology. In the past, two different results had been presented; one supporting using the root and another in favour of the stem. We have studied the effect of morphology on the performance of Arabic Information Retrieval and shown that the stem was the optimal indexing term in relation to the retrieval precision.

Furthermore, we have demonstrated in this thesis with empirical findings that in Arabic, as with other highly inflective languages, different morphemes can be combined to form a word, and thus it is the task of the stemming module to strip off all extra morphemes so that a stem or a root can be extracted for indexing. The one major contribution of the thesis to the advancement of Arabic Computational linguistics was the idea that a root solution can be developed with fewer morphological resources than was previously thought.

Every prior study in the field had its own tools, including tokenisers and stemmers, which were developed particularly for the task at hand except for two root stemmers: Khoja and Sebawai. They had been deployed in other AIR experiments conducted at a variety of institutions, and proved their robustness within IR applications. This thesis proposed a third root-finding approach that outperformed Khoja and Sebawai while using a smaller lexicon. Our algorithm recorded a 13% relative gain in terms of precision over Khoja and Sebawai.

The Simple Arabic Stemmer (SAS) presented in this thesis has contributed to the solution of the over-stemming problem by suggesting the enforcement of regulations on the segmentation process so that root radicals can be left intact. This was accomplished by creating a compatibility table that dictates the relationships amongst prefixes, patterns, and suffixes. In the Arabic language, there is a rule system defining what morphemes can be employed in the generation process. Through studying the Quran morphological content, we were able to infer the guidelines for determining the borderlines between different morphemes, and that has led to the inception of a more efficient root-finding algorithm with fewer lexical items.

References

[1] Sabah Al-Fedaghi and Fawaz Al-Anzi. A new algorithm to generate Arabic root-pattern forms. In *Proceedings of the 11th National Computer Conference and Exhibition*, pages 391–400, Dhahran, Saudi Arabia, March 1989. 40

[2] Ibrahim Al-Kharashi and Martha Evens. Comparing words, stems, and roots as index terms in an Arabic information retrieval system. In *Journal of the American Society for Information Science*, volume 45, pages 548–560. John Wiley Sons Inc., New York, Sept. 1994. 119

[3] Riyadh Al-Shalabi. *Design and Implementation of an Arabic Morphological System to Support Natural Language Processing*. PhD thesis, IIT, Chicago, Illinois, 1996. 40

[4] Imad Al-Sughaiyer and Ibrahim Alkharashi. Arabic morphological analysis techniques: A comprehensive survey, 2004. 6

[5] Mohammed Algarni, Brent Martin, Tim Bell, and Kourosh Neshatian. Simple Arabic Stemmer. In *the 23rd ACM International Conference on Information and Knowledge Management CIKM'14*, pages 1803–1806, Shanghai, China, November 2014. 65, 126

[6] Mohammed Aljlayl, Steve Beitzel, Eric Jensen, Abdur Chowdhury, D. Holmes, M. Lee, David Grossman, and Ophir Frieder. IIT at TREC-10. In *TREC-10*, Gaithersburg, Maryland, 2001. 52, 120, 122

[7] Mohammed A. Aljlayl. On Arabic Search: Improving the Retrieval Effectiveness via a Light Stemming Approach. In *the 11th ACM International*

Conference on Information and Knowledge Management CIKM'02, pages 340–347, Virginia, USA, November 2002. 122

[8] Mohammed A. Aljlayl. *On Arabic Search: The Effectiveness of Monolingual and Bidirectional Information Retrieval*. PhD thesis, Illinois Institute of Technology, Chicago, Illinois, 2002. 13, 139

[9] Ibrahim Alkharashi. *A Microcomputer-Based Arabic Information Retrieval System Comparing Words Stems Roots as Index Terms*. PhD thesis, Illinois Institute of Technology, Chicago, Illinois, May 1991. 13, 119

[10] Kenneth Beesley. Arabic Morphology Using Only Finite-State Operations. In *COLING-ACL'98 Proceedings of the Workshop on Computational Approaches to Semitic languages*, pages 50–57, 1998a. 43, 45

[11] Kenneth Beesley. Finite-State Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001. In *EACL 2001 Workshop Proceedings on Arabic Language Processing: Status and Prospects*, pages 1–8, Toulouse, France, 2001. 43

[12] Kenneth Beesley, Timothy Buckwalter, and Stewart Newton. Two-Level Finite-State Analysis of Arabic Morphology. In *The Seminar on Bilingual Computing in Arabic and English*, University of Cambridge, Cambridge, England, 1989. 13, 41, 42, 52

[13] Kenneth Beesley and Lauri Karttunen. *Finite State Morphology*. CSLI Studies in Computational Linguistics. CSLI Publications, Stanford, California, 2003. 65

[14] Michael Brame. *Arabic Phonology: Implications for Theory and Historical Semitic*. PhD thesis, MIT, Cambridge, Massachusetts, 1970. 6, 7, 11, 22, 23

[15] Chris Buckley and Ellen Voorhees. Evaluating evaluation measure stability. In *SIGIR*, pages 33–40, 2000. 118

[16] Tim Buckwalter. Lexicographic Notation of Arabic Noun Pattern Morphemes and their Inflectional Features. In *The Second Cambridge Conference*

on Bilingual Computing in Arabic and English,, Cambridge, England, 1990. 45, 52

[17] Timothy Buckwalter. Buckwalter Arabic Morphological Analyzer Version 2.0 Linguistic Data Consortium, Catalog Number LDC2004l02. Linguistic Data Consortium, 2004. xvi, 2, 13, 41, 52

[18] Aitao Chen and Fredric Gey. Building an Arabic Stemmer for Information Retrieval. In *TREC-11*, Gaithersburg, Maryland, 2002. NIST. 120

[19] Kareem Darwish. Building a Shallow Arabic Morphological Analyzer in one Day. In M. Rosner and S. Wintner, editors, *Computational Approaches to Semitic Languages, an ACL'02 Workshop*, pages 47–54, Philadelphia, PA, July 2002. Association for Computational Linguistics. 41, 52, 120

[20] Kareem Darwish. *Probabilistic Methods for Searching OCR-Degraded Arabic Text*. PhD thesis, University of Maryland, College Park, Maryland, 2003. 13, 41, 50, 115, 116

[21] Kareem Darwish and Ahmed Ali. Arabic retrieval revisited: Morphological hole filling. In *the 50th Annual Meeting of the Association for Computational Linguistics (ACL'12)*, volume 2, pages 218–222, Stroudsburg, PA, 2012. Association for Computational Linguistics. 9

[22] Kareem Darwish, Hany Hassan, and Ossama Emam. Examining the effect of improved context sensitive morphology on Arabic information retrieval. In *ACL-2005 Workshop on Computational Approaches to Semitic Languages*, Stroudsburg, PA, 2005. Association for Computational Linguistics. 10

[23] Kareem Darwish and Walid Magdy. *Arabic Information Retrieval*. Now Publishers, February 2014. 4

[24] Mona Diab. Second generation AMIRA tools for arabic processing: Fast and robust tokenization, pos tagging, and base phrase chunking. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, 2009. 41, 62

[25] Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. Automatic tagging of arabic text: From raw text to base phrase chunks. In *Proceedings of North American Association for Computational Linguistics*, pages 149–152, 2004. 62

[26] Judith Dror, Dudu Shaharabani, Rafi Talmon, and Shuly Wintner. Morphological Analysis of the Qur'an. *Literary and Linguistic Computing*, 19(4):431–452, 2004. 65, 67

[27] Kais Dukes and Nizar Habash. Morphological Annotation of Quranic Arabic. In *Language Resources and Evaluation Conference (LREC)*, Valletta, Malta, 2010. 68

[28] Bilel Elayeb and Ibrahim Bounhas. Arabic cross-language information retrieval: A review. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 15(18), 2016. 4

[29] Fredric C. Gey and Douglas W. Oard. The TREC-2001 Cross-Language Information Retrieval Track: Searching Arabic using English, French or Arabic Queries. In *TREC-10*, Gaithersburg, Maryland, 2001. 12, 112, 114

[30] David Graff and Kevin Walker. Arabic Newswire Part 1 Corpus (1-58563-190-6). Linquistic Data Consortium (LDC), 2001. 50

[31] Nizar Habash. Introduction to Arabic Natural Language Processing. In Graeme Hirst, editor, *Synthesis Lectures on Human Language Technologies*, volume 3, pages 1–87. Morgan & Claypool Publishers, August 2010. 53

[32] Nizar Habash and Owen Rambow. Arabic tokenization, part-of- speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the Conference of American Association for Computa- tional Linguistics*, 2005. 6, 62

[33] Nizar Habash, Owen Rambow, and Ryan Roth. MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS

tagging, stemming and lemmatization. In *Proceedings of the 2^{nd} International Conference on Arabic Language Resources and Tools (MEDAR)*, Cairo, Egypt, 2009. 41, 62

[34] Nizar Habash and Ryan Roth. Catib: the Columbia Arabic Treebank. In *Association of Computational Linguistics (ACL'09)*, Suntec, Singapore, August 2009. 68

[35] David A. Hull. Using statistical testing in the evaluation of retrieval performance. In *Proceedings of the 16^{th} ACM SIGIR*, pages 329–338, 1993. 124, 141

[36] David A. Hull. Stemming Algorithms - A Case Study for Detailed Evaluation. *Journal of the American Society for Information Science. Special Issue: Evaluation of Information Retrieval Systems*, 47(1):70–84, 1996. 64

[37] Ronald Kaplan and Matin Kay. Pholonolgical Rules and Finite-State Transducers. In *The annual meeting of the Linguistic Society of America*, New York City, USA, 1981. 22, 43

[38] Shereen Khoja and Roger Garside. Stemming Arabic Text. *Computing Department, Lancaster University*, 1999. 13, 41, 46

[39] George Kiraz. Multi-Tape Two-Level Morphology: A case study in semitic non-linear morphology. In *COLING'94*, volume 1, pages 180–186, 1998. 5

[40] Kimmo Koskenniemi. *Two-Level Morphology:A General Computational Model for Word-Form Recognition and Production*. PhD thesis, University of Helsinki, Finland, 1983. 43

[41] Robert Krovets. Viewing Morphology as an Inference Process. In *the 16^{th} International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, USA, 1993. 33

[42] Klaus Lagally. ArabTEX: Typesetting Arabic and Hebrew, User Manual Version 4.00. Technical Report 2004/03, Fakultät Informatik, Universität Stuttgart, 2004. 2

[43] Leah Larkey, Lisa Ballesteros, and Margaret Connell. Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis. In *Proceedings of the 25th ACM SIGIR*, pages 275–282, Tampere, Finland, August 2002. ACM. 41, 122

[44] Leah Larkey and Margaret Connell. Arabic Information Retrieval at UMASS. In *TREC-10*, Gaithersburg, Maryland, 2001. 13, 52, 58, 120, 122

[45] Mary Levy. *The Plural of the Noun in Modern Standard Arabic*. PhD thesis, The University of Michigan, Ann Arbor, Michigan, 1971. 32, 91, 94

[46] Al-Areeb Electronic Publishers LLC. Electronic copy of zad almEAd fy hdy xyr AlEbAd. 116

[47] Mohamed Maamouri and Bies Ann. Developing an Arabic Treebank: Methods, Guidelines, Procedures, and Tools. In *COLING 2004 Workshop on Computational Approaches to Arabic Script-based Languages*, pages 2–9, 2004. 53, 68

[48] Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The penn Arabic treebank: Building a large-scale annotated arabic corpus. In *the NEMLAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, 2004. 53, 62

[49] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *An Introduction to Information Retrieval*. Cambridge University Press, Cambridge, England, 2009. 112

[50] John McCarthy. *Formal Problems in Semitic Phonology and Morphology*. PhD thesis, MIT, Cambridge, Massachusetts, 1979. 5, 6, 26

[51] John Mccarthy. A Prosodic Account of Arabic Broken Plurals. In *Linguistics Department Faculty Publication Series*, pages 289–320. University of Massachusetts, Amherst, Massachusetts, 1983. 32, 91

[52] John McCarthy and Alan Prince. Foot and Word in Prosodic Morphology: The arabic broken plural. In *Natural Language & Linguitics Theory*, volume 8, pages 209–282. Springer, May 1990. 5

[53] Haidar Moukdad. *A Comparison of Root and Stemming Techniques for the Retrieval of Arabic Documents*. PhD thesis, McGill University, Montreal, Canada, 2001. 139

[54] Ajit Narayanan and Lama Hashem. On abstract finite-state morphology. In *Proceedings of the Sixth Conference on European Chapter of the Association for Computational Linguistics*, pages 297–304, Stroudsburg, PA, USA, 1993. Association for Computational Linguistics. 136, 145

[55] Alexis Neme and ric Laporte. Pattern-and-Root Inflectional Morphology: the Arabic Broken Plural. In *Language Sciences*, volume 40, pages 221–250. Elsevier, 2013. 32, 91

[56] Craig Nevill-Manning and Ian Witten. Identifying Hierarchical Structure in Sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, 1997. 71

[57] Muhammad FwaAd Abdull-Baqy (محمد فؤاد عبدالباقي). *The Indexed Lexicon of the Quran Words (المعجم المفهرس لألفاظ القرآن الكريم)*. DAr Alkutub AlmaSryp (دار الكتب المصرية) [Egyptian Books House], Cairo, Egypt, 1364 H. 66

[58] Abdusalam F Ahmed Nwesri. *Effective Retrieval Techniques for Arabic Text Effective Retrieval Techniques for Arabic Text*. PhD thesis, RMIT University, Melbourne, Victoria, Australia, May 2008. 120

[59] Douglas W. Oard and Fredric C. Gey. The TREC-2002 Arabic/English CLIR Track. In *TREC-11*, Gaithersburg, Maryland, 2002. 12, 115

[60] AHmad MuxtAr Omar and Team (أحمد مختار عمر بمساعدة فريق عمل). *The Lexicon of the Quran Words (المعجم الموسوعي لألفاظ القرآن الكريم وقراءاته)*. sTwr AlmErfh (سطور المعرفة) [The Knowlege Rows], Riyadh, Saudi Arabia, 2002. 77

[61] Arfath Pasha, Mohamed Al-Badrashiny, Ahmed ElKholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Language Resources and Evaluation Conference (LREC)*, 2014. 4, 62

[62] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980. 2

[63] Ibn Manzwr (ابن منظور). *LisAnu AlEarab (لسان العرب) [The Arabs Tonque]*. Dar AlmEarf (دار المعارف) [Knowledge House], Cairo, Egypt, 2003. 28

[64] Alxalyl Bin AHmad (الخليل بن أحمد الفراهيدي). *KitAbu AlEyn (كتاب العين) [The Ain Book]*. Dar Alkutub AlElmyh (دار الكتب العلمية) [Scientific Books House], Beirut, Lebanon, 2003. 27

[65] AbdullRahman AlAnbAry (عبدالرحمن الأنباري). *AsrAru AlEarabyp (أسرار العربية) [The Arabic Secrets]*. Dar Alkutub AlElmyh (دار الكتب العلمية) [Scientific Books House], Beirut, Lebanon, 1997. 17

[66] Bdr Al-Dyn Alzarkashy (بدر الدين محمد بن عبدالله الزركشي). *Alburhan fi*

Elwm AlQuraAn (البرهان في علوم القرآن). Dar AlturAth [Heritage House] (دار التراث), Cairo, Egypt, 1984. 17

[67] Anne De Roeck and Waleed Al-Fares. A morphologically sensitive clustering algorithm for identifying arabic roots. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics ACL'00*, pages 199–206, Hong Kong, 2000. Association for Computational Linguistics. 40

[68] Majdi Shaker Salem Sawalha. *Open-source Resources and Standards for Arabic Word Structure: Fine Grained Morphological Analysis for Arabic Text Corpora*. PhD thesis, University of Leeds, Leeds, England, 2011. 139

[69] AlHusyn Bin AHmad Alzawzany (الحُسَين بن أَحمَد الزَّوزنيّ). *The Seven Poems Explanation (شَرح المُعَلَّقَات السَّبع)*. Dar AlmErfp (دَار المَعرفَة) [Knowledge House], Beirut, Lebanon, 2004. 16

[70] Othman Bin Jenny (عُثمَان بن جنّي النحوي). *AlmunSif (المُنصِف) [The Unbiased]*. Ministry of Public Knowledge (وَزَارة المَعَارف العمومية), Cairo, Egypt, 1954. 26, 28, 96

[71] Othman Bin Jenny (عُثمَان بن جنّي النحوي). *AlxaSaAyS (الخَصَائص) [The Attributes]*. (دَار الكتب العلمية) [Scientific Books House], Beirut, Lebanon, 2001. 26

[72] Otakar Smrž. *Functional Arabic Morphology. Formal System and Implementation*. PhD thesis, Charles University in Prague, Prague, Czech Republic, 2007. 37

[73] Otakar Smrẑ and Jan Hajiĉ. The Other Arabic Treebank: Prague Dependencies and Functions. In *Arabic Computational Linguistics: Current Implementations*. CSLI Publications, 2006. 68

[74] Abdelhadi Soudi, Antal van den Bosch, and Gunter Neumann. *Arabic Computational Morphology*, volume 38 of *Text, Speech and Language Technology*. Springer, Dordrecht, The Netherlands, 2007. 4, 13, 18

[75] Kazem Taghva, Rania Elkhoury, and Jeffrey Coombs. Arabic stemming without a root dictionary. In *the International Conference on Information Technology: Coding and Computing (ITCC'05)*, volume 1, pages 152–157. IEEE, April 2005. 40

[76] Rafi Talmon and Shuly Wintner. Morphological Tagging of the Qur'an. In *Workshop on Finite-State Methods in Natural Language Processing, an EACL'03 Workshop*, Budapest, Hungary, 2003. 67

[77] `http://www.internetlivestats.com`. Internet live stats 1/7/2014. 12

[78] `http://www.merriam-webster.com`. Merriam-webster online 23/8/2014. 22

[79] Ellen Voorhees and Donna Harman (eds.). *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, Cambridge, Massachusetts, 2005. 118

[80] Thomas H. Wonnacott and Ronald J. Thomas H. Wonnacott. *Introductory Statistics*. John Wiley & Sons Inc., New York, USA, 1990. 124, 141

[81] William Wright. *A Grammar of the Arabic Language*. Cambridge University Press, Cambridge, England, 1971. 23, 27, 28, 30, 84, 86, 90, 98